**Politechnika Świętokrzyska**

**Wydział Elektrotechniki, Automatyki i Informatyki**

Michał Mazur

**Numer albumu: 70043**

**Zastosowanie chmury na przykładzie aplikacji pośrednictwa pracy**

Praca dyplomowa magisterska

na kierunku Informatyka

Opiekun pracy dyplomowej:

Dr inż. Barbara Łukawska

Zakład Informatyki

Kielce, 2017

**Streszczenie**

Głównym celem pracy magisterskiej jest stworzenie aplikacji webowej przeznaczonej dla programistów poszukujących pracy oraz pracodawców oferujących zatrudnienie. Aplikacje napisano w technologii Asp .NET Core, w środowisku Microsoft Visual 2017, wykorzystano framework Bootstrap, jQuerry, a do autoryzacji użytkowników zastosowano narzędzie Asp .NET Identity 2. Aplikacja wraz z bazą danych MSSQL została umieszczona w chmurze obliczeniowej Microsoft Azure. Serwis oferuje przeglądanie ofert pracy, przeglądanie zarejestrowanych programistów, aplikowanie na wybrane stanowiska, składanie ofert programistom przez pracodawców oraz negocjowanie warunków zatrudnienia.

**Abstract**

The main purpose Master's thesis is creation a web application for developers and employers offering employment. Applications written in Asp .NET Core technology, using the Microsoft Visual Studio 2017 framework, use the Bootstrap framework, jQuerry, and Asp .NET Identity 2 for user authentication. The application, along with the MSSQL database, was placed in the Microsoft Azure cloud. The service offers job viewing, browsing of registered developers, applying for job, submitting offer to developers by employers, and negotiating employment terms.

# Wstęp

Chmura to jeden z trendów, który rozwija się bardzo dynamicznie w świecie IT. Chmura oferuje szerokie możliwości, od przechowywania plików, prostych serwisów internetowych, baz danych, aplikacji: np Microsoft PowePoint czy Skype mają swój „chmurowy” odpowiednik, po przez rozbudowane i potężne systemy ERP. Chmury obliczeniowe zyskują coraz większą popularność ze względu na oferujące przez siebie liczne zalety i możliwości. Chmura obliczeniowa to bardzo wygodne rozwiązanie. Usługodawca umożliwia wykorzystanie infrastruktury i jej mocy obliczeniowej, w zamian za co miesięczne opłaty. Jej ogromną zaletą jest elastyczność. Usługobiorca płaci za realnie wykorzystaną moc obliczeniową i w każdej chwili, gdy potrzebuje większej mocy, natychmiastowo ją dostaje. To ogromna przewaga na wirtualnymi prywatnymi serwerami, gdzie udostępnione zasoby są stałe przez cały czas trwania umowy. Często zdarzają się wiec sytuacje, gdzie zasoby są marnowane lub gdzie aplikacja przestaje działać ze względu na przeciążenia wynikające, ze zbyt dużego ruchu w serwisie.

W ramach pracy magisterskiej zostały omówione i przybliżone zagadnienie chmury obliczeniowej, jej cechy, rodzaje oraz zastosowanie. Istnieje wiele usługodawców oferujących przetwarzanie w chmurze jak np.: Microsoft, Google, Amazon czy polski Oktawave, jednakże na potrzebny pracy magisterskiej wykorzystano chmurę obliczeniową Microsoft Azure na zasadzie licencji studenckiej DreamSpark. Temat pracy magisterskiej to „Zastosowanie chmury na przykładzie aplikacji pośrednictwa pracy”. Przedstawia możliwości wykorzystanej chmury obliczeniowej Microsoft Azure i w ramach pracy magisterskiej została opracowana i wdrożona aplikacja webowa dla programistów oraz potencjalnych pracodawców, która umożliwia wyszukiwanie ofert pracy przez programistów oraz wyszukiwanie potencjalnych kandydatów do pracy przez programistów. Z pośród wielu technologii, zdecydowano, ze względu na elastyczność, międzyplatformowość i przede wszystkim możliwości, wykorzystać wydany w 2016 roku przez Microsoft najnowszy i rewolucyjny framework Asp .NET Core. Dotychczas głównym minusem tej technologii, było jej ograniczenie tylko do platformy Microsoftu, jednak po rewolucyjnych zmianach .NET Core ma szanse pokonać technologie, które dotychczas były darmowe i multiplatformowe, jak chociażby Java czy PHP.

# Chmura – wprowadzenie

Pojęcie chmury w świecie IT, istnieje już od kilku dobrych lat. Jest jednym, z trenów, który rozwija się bardzo dynamicznie. Analitycy przewidują, że to właśnie tutaj, ukryty jest ogromny potencjał, który może zostać wykorzystany i który może ukształtować kolejne lata w IT. Ogólnie rozumianą „chmurę” można podzielić na dwie kategorie. Wyróżniamy chmurę publiczną, z którą ma odczynienia większość zwykłych użytkowników, czy to komputerów, laptopów, smartfonów lub innych urządzeń mobilnych. Często zdarzają się sytuacje, że brakuje miejsca na lokalnym dysku naszego urządzenia, więc część danych można przenieść na dyski chmurowe. Użytkownicy, mogą skorzystać w tedy, z ofert takich firm jak np.: Microsoft, Google czy nawet Dropbox, założyć darmowe lub płatne konto w ich serwisie i wrzucić swoje pliki na ich przestrzeń dyskową. Rozwiązanie bardzo wygodne i nowoczesne, nie trzeba inwestować w nowe urządzenia z większą pamięcią czy dokupywać zewnętrzne dyski, wystarczy wszystko przenieść do chmurę i mieć do niej dostęp z dowolnego miejsca na świecie, z dowolnego urządzenia. Problemem, może okazać się przestrzeń dyskowa, w podstawowych, bezpłatnych wersjach możemy liczyć na maksymalnie 15 GB przestrzeni dyskowej (w przypadku Google Drive). Drugim typem chmury, który zostanie dokładniej omówiony i przetestowany w ramach pracy magisterskiej jest chmura obliczeniowa (ang. Cloud computing)

## Cloud-Computing.jpg (683×421)

Rysunek 2.1 Cloud Computing

<http://www.salesforcetutorial.com/wp-content/uploads/2013/04/Cloud-Computing.jpg>

dostęp z dnia 05.05.2017

## Definicja chmury obliczeniowej

Definicja chmury obliczeniowej, którą można znaleźć w słowniku pojęć Głównego Urzędu Statystycznego [1]:

„Usługi chmury obliczeniowej to możliwość korzystania ze skalowalnych usług ICT przy zastosowaniu Internetu. Usługi świadczone w chmurze obliczeniowej, mogą obejmować dostęp do oprogramowania, korzystanie z określonej mocy obliczeniowej, przechowywanie danych. Wymienione usługi dostarczane są przy wykorzystaniu serwerów usługodawcy, mogą być skalowalne w górę lub w dół (skalować można na przykład liczbę użytkowników czy też ilość przechowywanych danych, -mogą być zmieniane lub dostosowywane na własne żądanie (ang. on-demand self-service) tzn. bez konieczności ingerencji usługodawcy, mogą podlegać formom płatności takim jak abonament lub płatność elastyczna dostosowana do ilości i rodzaju zakupionych usług.”

Ta definicje dość dobrze obrazuje, czym jest i jakie usługi mogą być świadczone w ramach cloud computing. Jednak dla zwykłego użytkownika, może to określenie okazać sie zbyt profesjonalne i niezrozumiałe. Tłumacząc sformułowanie dotyczące chmury obliczeniowej, na język bardziej zrozumiały, chmura obliczeniowa, to wykorzystanie pewnych zasobów pamięci, procesora, przestrzeni dyskowej, zdalnie, czyli przez sieć LAN lub przez Internet. Jednakże, to tylko część definicji, określającej przetwarzanie w chmurze. Aby, nazwać coś przetwarzaniem w chmurze, należy dostarczyć usługę, a nie gotowy produkt. Te dwa aspekty, czyli zdalne wykorzystanie zasobów w ramach dostarczonych usług, sprawia, że mówimy o przetwarzaniu w chmurze [2]. Obrazując to, czym jest usługa, a gotowy produkt, najlepiej posłużyć się przykładem. Usługą może być prąd, który jest dostarczany przez medium transmisyjne jakim jest sieć elektryczna. Z punktu widzenia odbiorcy, czyli każdego użytkownika, który podłącza urządzenie elektryczne do gniazdka, nie ma znaczenia, jak zbudowana jest cała infrastruktura, gdzie znajduje się elektrownia, czy występują jakieś przeciążenia na liniach wysokiego i średniego napięcia. Najważniejsze to to, aby cały czas było napięcie w gniazdku sieci elektrycznej. Gdyby odbiorca kupił generator prądu elektrycznego i zasilał całego gospodarstwo domowe na własną rękę, można by powiedzieć, że to jest właśnie gotowy produkt. W tedy, na barki klienta spada odpowiedzialność za całą infrastrukturę, kupno odpowiedniego paliwa do generatora, regularne sprawdzanie poziomu tego paliwa czy serwisowanie generatora. Nie można wykluczyć również, prawdopodobieństwa wymiany, generatora w przypadku jego awarii. Przenosząc to, na realia przetwarzania chmurowego, standardowym podejściem było by, kupienie gotowego produktu, a więc wykupienie przestrzeni dyskowej, czy całego serwera u jakiegoś dostawcy. Chmurowe podejście to wykupienie usługi o określonych parametrach. Nie interesuje nas, w tedy gdzie fizycznie nasze pliki są przechowywane, na jakich maszynach są przetwarzane, to wszystko ma nam zagwarantować dostawca. To on, odpowiada za takie aspekty jak bezpieczeństwo, dostępność, szybkość czy stabilność. Podsumowując – chmura, to zasoby obliczeniowe, które są rozproszone, do których dostęp określają usługi, komputery klienckie posiadają stałe łącze do zasobów i mogą z nich w każdej chwili skorzystać.

## Krótka historia

Pierwszy raz stwierdzenie, które jest bliskie idei cloud computing, zostało sformułowane w 1960 roku, „obliczenia mogą kiedyś być zorganizowane jako usługa użyteczności publicznej” autorem tych słów jest John McCarthy. Prawdopodobnie, termin cloud computing, wywodzi się od schematów chmur, które były używane do reprezentacji Internetu na różnych schematach. W latach 50-tych, można już doszukać się rozwiązań, które przypominają rozwiązania stosowane w przetwarzaniu chmurowym. W tamtych latach, zasoby komputerów typu mainframe były udostępniane lokalnym komputerom-klientom, których moc obliczeniowa była znikoma. Na rewolucje, trzeba było czekać, aż do lat 90-tych, firmy z branży telekomunikacyjnej, zaczęły udostępniać wirtualne sieci prywatne (VPN), co przyczyniło się później do rozwoju chmury. W 1997 roku, pierwszy raz we współczesnym IT, podczas wykładu Ramnatha Chellappa został użyty zwrot cloud computing. Określono nowy paradygmat – granica usług, obliczeniowych zostanie ustalona przez uzasadnienie ekonomiczne, a nie ograniczenia technologiczne. Pierwszy etap rozwoju przetwarzania chmurowego, zawdzięczamy firmie salesforce.com [3]. Zajmowała się ona, dostarczaniem aplikacji biznesowych, za pośrednictwem strony www. Była pionierem, w dziedzinie dostarczania usług przez Internet. Koleny krok, to firma Amazon. W 2002 roku wprowadziła usługi cloud, czyli magazynowanie i przeprowadzanie pewnych operacji dla usług takich jak Amazon Mechanical Turk – są to usługi pozwalające wykonywać określone czynności ludziom, czynności, które nie mogą zostać zautomatyzowane. Cztery lata później Amazon uruchomił EC2 (Elastic Compute Cloud), usługę pozwalającą wynajmować prywatne serwery klientom, którzy to, byli w stanie uruchamiać na nich własne aplikacje. Przełomowym rokiem okazał się rok 2009. Dwaj giganci w świecie IT, Google oraz Microsoft otworzyli swoje własne serwisy chmurowe. Google zaczął oferować Google Apps, czyli aplikacje oparte na przeglądarce, natomiast Microsoft uruchomi wersje beta Windows Azure. Uruchomienie Azure’a, jest jednym z istotniejszych wydarzeń w historii cloud computing. Aktualnie, coraz więcej przedsiębiorstw decyduje się na przeniesienie swojej infrastruktury do chmury. Jak wynika z badań przeprowadzonych we Francji, Hiszpanii, Holandii, Niemczech, Polsce, Rosji, Włoszech i Wielkiej Brytanii na zlecenie firmy VMware, już **60% europejskich firm,** przeniosło część swojej infrastruktury informatycznej do chmury. Sytuacja w Polsce, jest odrobine gorsza, bo z usług w chmurze korzysta **46% polskich przedsiębiorstw**z sektora małych i średnich przedsiębiorstw [4].

## Podstawowe pojęcia związane z chmurą obliczeniową

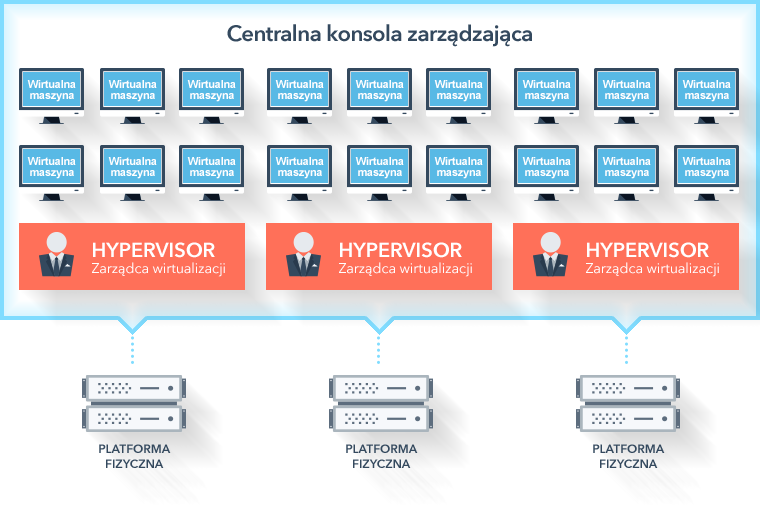
### Wirtualizacja

Istotną kwestią, jeśli chodzi o zrozumienie działania chmury obliczeniowej, jest zrozumienie czy jest wirtualizacja. Wirtualizacja, inaczej rozumiana jako konsolidacja, to technologia, dzięki której, administrator jest wstanie efektywnie wykorzystać sprzęt, do uruchomienia wielu systemów operacyjnych czy kopi oprogramowania. To wszystko, jest możliwe dzięki udostępnianiu i przydzielaniu komputera lub serwera [5]. To pewnego rodzaju emulacja, system operacyjny ma wrażenie, że działa na fizycznym sprzęcie, a w rzeczywistości pracuje w maszynie wirtualnej, której udostępniono jakąś część zasobów. Takie rozwiązanie ma ogromną zaletę – oszczędność. Można uruchomić jedną maszynę, zamiast pięciu i tak podzielić jej zasoby, aby istniała możliwość, uruchomienia pięciu zwirtualizowanych środowisk. Takie rozwiązanie, pozwala na lepszą utylizację zasobów oraz uniezależnia środowisko uruchomieniowe od konkretnej maszyny fizycznej. Dodatkowo, jest mniej podatne na awarie czy usterki. Awaria sprzętu, nie doprowadzi do awarii maszyny wirtualnej czy systemu operacyjnego. W przypadku usterki bądź awarii, zachowując ciągłość pracy, można daną maszynę wirtualną przenieść do innego zasobu fizycznego. Po skonfigurowaniu i uruchomieniu maszyn wirtualnych na określonym zasobie fizycznym, maszyny cechuje elastyczność. W dowolnym momencie, bez problemu można przydzielić danej maszynie większą ilość np.: pamięci ram czy miejsca na dysku twardym. Aby, proces wirtualizacji przebiegał prawidłowo, potrzebne jest to tego specjalne oprogramowanie. Hipernadzorca (ang. Hypervisor) czuwa nad prawidłowym wirtualizowaniem zasobów [6].

Na rynku najczęściej spotykane hipervisor’y to:

* Xen
* Microsoft Hyper-V
* Vmware ESXi

Wirtualizacja zasobów została przedstawiona na Rysunek 2.2



## Rysunek 2.2 Wirtualizacja zasobów

## http://www.decsoft.com.pl/userfiles/tinyMCE/wirtualizacja\_zasobow\_01.png

W kontekście chmury obliczeniowej, najczęściej wirtualizowane są serwery. Zastępuje się serwery, których moc obliczeniowa jest słabo wykorzystywana, większą ilością mniejszych, zwirtualizowanych serwerów, które lepiej wykorzystują zasoby fizycznych serwerów. Konsolidowane, są ogromne ilości rozproszonych, fizycznych maszyn, które są przekształcane w jedną wirtualną warstwę. Klient końcowy, który korzysta z chmury obliczeniowej, dostaje platformę, w której w prosty sposób, może zarządzać swoimi aplikacjami i może dynamicznie regulować moc obliczeniową, dla jego aplikacji. To wszystko, jest możliwe, dzięki potędze jaka kryje się w wirtualizacji.

### Serwer wirtualny

W poprzednim rozdziale dokładnie została opisania wirtualizacja, jednakże, nie zostało dobrze wytłumaczone czym, tak naprawdę, jest serwer wirtualny. Serwer wirtualny to warstwa pośrednia, pomiędzy fizycznym sprzętem, a środowiskiem uruchomieniowym, w którym oprogramowanie, odwołuje się do sprzętu lub do systemu operacyjnego. Udostępnia, w sposób zwirtualizowany, to co udostępniałby fizyczny serwer dla aplikacji [7]. Z punktu widzenia użytkownika, nie ma żadnej różnicy, między serwerem fizycznym, a wirtualnym. Bez problemu, użytkownik jest wstanie przenieść aplikacje z jednego serwera, na drugi. Jednak, należy podkreślić, że serwer wirtualny to tylko pamięć RAM. Dane są przechowywane na storage’u blokowym.

### Storage blokowy

Storage blokowy to wydzielony obszar na fizycznym dysku twardym, który pozwala odczytywać i zapisywać dane w postaci bloków. Jest odpowiednikiem zwykłego komputerowego dysku twardego. Bardzo wydajny w sytuacjach, gdy często trzeba odczytywać, modyfikować i zapisywać dane np.: z bazy danych. Storage blokowy, należy dołączyć, do uruchomionego serwera wirtualnego, aby móc na nim zainstalować aplikacje oraz system operacyjny.

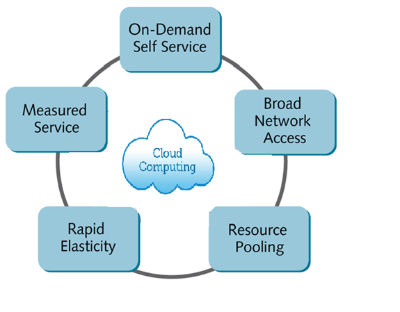
### Storage obiektowy

Wielu dostawców, oferuje oprócz dysków blokowym – które, przechowują system operacyjny oraz bazę danych – dyski obiektowe. Storage obiektowy, magazynuje dowolne obiekty, które są hierarchizowane, do struktur typu drzewa czy listy. Można zapisywać do niego całe obiekty, nie patrząc na rozmiar tych obiektów. Świetnie sprawdza się w przypadku przechowywania zdjęć czy plików wideo oraz innych statycznych plików. W dysku obiektowym, liczy się możliwość taniego i wydajnego przechowywania dużej ilości obiektów, podczas gdy w storage’u blokowym najważniejsza jest szybkość.

# Charakterystyka chmury obliczeniowej

## Cechy

Istnieje pięć podstawowych cech modelu cloud computing, które zostały zdefiniowane przez Narodowy Instytut Norm i Technologii (NIST), a następnie zostały redefiniowane przez dużą liczbę architektów i ekspertów [8]. Zgodnie z informacjami NIST, cloud computing jest modelem umożliwiającym wszechobecny, wygodny, dostęp do sieci na żądanie, dostępnej z wspólnej puli konfigurowalnych zasobów obliczeniowych (np. Sieci, serwerów, pamięci masowej, aplikacji i usług), które mogą być szybko udostępniane, ponosząc minimalny wysiłek włożony w zarządzanie usługą lub interakcje z dostawcą usług. Ten model chmury, składa się z pięciu podstawowych cech [9] (Rysunek 3.1):



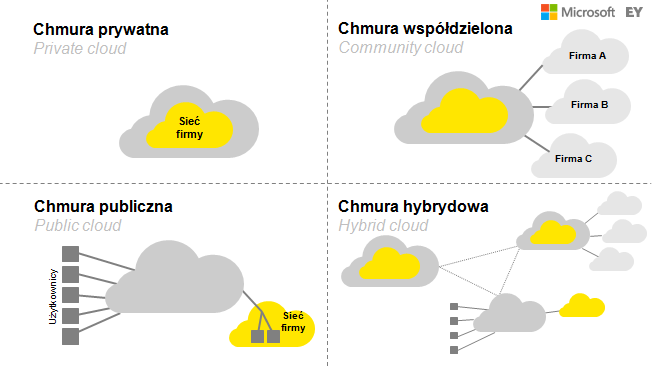
Rysunek 3.1 Cechy modelu cloud computing

<https://www.linkedin.com/pulse/five-essential-characteristics-cloud-computing-sankar-somepalle>

* Samodzielna obsługa na żądanie (ang. On-Demand Self Service) - konsumenci mogą udostępniać zasoby komputerowe, takie jak czas serwera i dysk sieciowy, bez potrzeby interakcji z dostawcą usług.
* Szeroki dostęp do sieci (ang. Broad Network Access) – możliwy jest dostęp, przez Internet, za pośrednictwem standardowych mechanizmów, które promują wykorzystanie, przez heterogeniczne (niejednorodne) platformy klienckie (np. Telefony komórkowe, tablety, laptopy i stacje robocze).
* Łączenie zasobów (ang. Resource Pooling) - zasoby obliczeniowe dostawcy, są gromadzone w celu obsługi wielu konsumentów, przy użyciu modelu wielorodzinnego, z różnymi fizycznymi i wirtualnymi zasobami, które są przydzielane dynamicznie, zgodnie z potrzebami konsumentów. Występuje niezależność lokalizacji, ponieważ klient, zazwyczaj nie ma żadnej kontroli, ani wiedzy na temat dokładnej lokalizacji udostępnionych zasobów, ale może określić położenie na wyższym poziomie abstrakcji (np. w danym państwie czy rejonie). Zasoby obejmują pamięć, przetwarzanie i przepustowość sieci.
* Duża elastyczność (ang. Rapid Elasticly) – możliwości dynamicznego skalowania zasobów są bardzo duże. W sposób bardzo elastyczny i nawet automatyczny można dostosowywać udostępnione zasoby do potrzeb klientów. Dla konsumenta, możliwości udostępniania często wydają się być nieograniczone i można je w każdej chwili przydzielić w dowolnej ilości.
* Mierzona usługa (ang. Measured Service) - chmura obliczeniowa, automatycznie kontroluje i optymalizuje wykorzystanie zasobów, dzięki możliwości pomiaru na pewnym poziomie abstrakcji, odpowiedniej do typu usługi (np. przechowywania, przetwarzania, szerokości pasma). Wykorzystanie zasobów może być monitorowane, kontrolowane i raportowane, zapewniając przejrzystość, zarówno dostawcy jak i konsumentowi.

## Rodzaje

Aby, dostosować się do różnych potrzeb użytkowników, opracowano cztery rodzaje chmur obliczeniowych. Rysunek 3.2 przedstawia wszystkie rodzaje obecnie stosowanych chmur obliczeniowych.



Rysunek 3.2 Rodzaje chmur obliczeniowych

http://docplayer.pl/docs-images/24/2838725/images/3-0.png

### Publiczna

Przede wszystkim, chmura publiczna przeznaczona jest, do korzystania, przez większą grupę użytkowników, odpłatnie lub też bezpłatnie. Każdy użytkownik, za pośrednictwem Internetu, ma nieograniczony dostęp do chmury. Może ułatwiać współprace osobom, które znajdują się w odległych miejscach na świecie. Użytkownik, może też udostępniać swoje rozwiązania innym użytkownikom w tej chmurze. Infrastruktura chmury publicznej, jest rozproszona w wiele miejsc na całym świecie czy też kontynencie. Klient, nie ma wiedzy, o infrastrukturze usługi czy też lokalizacji usługodawcy. Przykładem chmury publicznej, która jest dostępna dla każdego użytkownika z dostępem do Internetu, jest chmura publiczna Google’a, która udostępnia aplikacje Google Docs, a nawet przykładem chmury publicznej z których większość użytkowników może nie zdawać sobie sprawy, są skrzynki pocztowe takie jak m.in. Gmail.

### Prywatna

Różnica między prywatna, a publiczną chmura jest taka, że chmura prywatna jest przeznaczona dla ograniczonej liczby użytkowników (najczęściej dla pracowników danej organizacji), a infrastruktura jest obsługiwana tylko przez jedna organizacje. Infrastruktura, zależy więc wyłącznie od przedsiębiorcy. Wszystkie urządzenia fizyczne, które zasilają chmurę obliczeniową, musza być skonfigurowane przez tą organizację i są pod jej całkowitą kontrolą. Możliwości, w przypadku chmury prywatnej są większe. Pozwala chociażby, na wprowadzenie wewnętrznego procesu rozliczania z wykorzystanych zasobów, z jednoczesnym zapewnieniem wysokiej elastyczności i efektywności, ale niestety o aspekty związane z bezpieczeństwem trzeba zadbać samemu [10].

### Społecznościowa

Chmura społecznościowa, jest typem chmury z infrastrukturą przeznaczoną dla większej społeczności np.: kilku organizacji tworzących daną społeczność. Osoby zrzeszone w społeczność mają zazwyczaj wspólne cele i problemy. Przykładem chmury społecznościowej

jest stworzone przez Chrysler, Ford i General Motors, Automotive Composites Consortium w celu wymiany informacji w badaniach nad wykorzystaniem komponentów polimerowych [11].

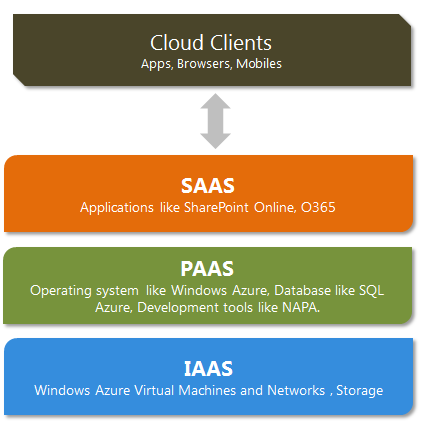
### Hybrydowa

Chmura hybrydowa, jest kombinacją chmury prywatnej i publicznej. Infrastruktura, składa się z dwóch lub więcej chmur (prywatnej i publicznej), które zachowują swoje unikalne cechy, ale pracują wspólnie, jako zespół. Jest do dobre rozwiązanie dla firm, które z jakiś przyczyn nie mogą sobie pozwolić na inwestycje w dużą i wydajną obliczeniową chmurę prywatną, a chcąc zachować wydajność i mieć w razie czego do dyspozycji większą moc obliczeniową, decydują się w momencie wykorzystania wszystkich prywatnych zasobów, część danych przenieść do chmury publicznej. Inną sytuacją może być bezpieczeństwo i kontrola nad poufnymi danymi. Dane poufne, można przechowywać we własnych prywatnych centrach danych, natomiast reszta zasobów, może być przechowywana i przetwarzana w chmurze publicznej.

## Modele

Ze względu na rodzaj świadczonych usług, można wyróżnić trzy modele (Rysunek 3.3):

* Oprogamowanie jako usługa (ang. Software as a Service, SaaS)
* Platforma jako usługa (ang. Platform as a Service, PaaS)
* Infrastruktura jako usługa (ang. Infractructure as a Service, IaaS)



Rysunek 3.3 Modele chmur obliczeniowych i przykłady ich zastosowań

http://www.cmswire.com/~/media/d01c36afeeae4036b43197a26c26932d.png

Wyznaczenie różnych modeli chmur ma pewną korzyść. Mogą one zbudować stos, w którym każda kolejna warstwa będzie realizowała swoje zadania i łączyła się ze swoimi sąsiednimi warstwami. W ramach każdej warstwy, działają usługi na wydzielonych dla tej warstwy zasobach.

### Oprogramowanie jako usługa (SaaS)

Oprogramowanie jako usługa, to chmura obliczeniowa, której dostawca udostępnia za pośrednictwem Internetu, dostęp do różnego rodzaju oprogramowania. Można to nazwać oprogramowaniem na żądanie. Cały - kolokwialnie mówiąc – silnik aplikacji, znajduje się w chmurze, dostawca udostępnia tylko interfejs i połączenie do aplikacji przez przeglądarkę [12]. Użytkownik, ma możliwość użytkowania aplikacji w przeglądarce internetowej. Rozwiązanie bardzo wygodne, nie niesie za sobą konieczności instalacji aplikacji na dysku swojego urządzenia, dodatkowo, jest możliwy dostęp z dowolnego urządzenia, z dowolnego miejsca na Ziemi. Często, dostęp do oprogramowania jest darmowy, jednakże zazwyczaj trzeba opłacać miesięczną stałą kwotę lub też w zależności od dostawcy, płaci się za wykorzystanie danego zasobu. Najczęściej, wykupuje się w ramach SaaS dostęp do całych pakietów narzędzi. Przykładem, może być pakiet narzędzi biurowych (edytor tekstu, arkusz kalkulacyjny, edytor prezentacji) oferowanych przez Google (Google Docs) czy Microsoft (Office 365). Większe przedsiębiorstwa korzystają również z bardziej zaawansowanych aplikacji jakimi są np.: systemy ERP.

### Platforma jako usługa (PaaS)

Platforma jako usługa, oferuje znacznie więcej niż Saas. Dostawca zapewnia dostęp, do całej platformy obliczeniowej, która oferuje biblioteki i narzędzia służące do tworzenia swoich własnych aplikacji [13]. Za pomocą tych narzędzi, można zbudować aplikacje, które następnie będzie można udostępnić w Internecie, jako swój serwis. Zwykle na takiej platformie jest już zainstalowany system operacyjny wraz z środowiskiem uruchomieniowym i bazą danych. Naliczanie opłat, za korzystanie z PaaS jest bardziej złożone niż w przypadku SaaS. Na całkowity miesięczny koszt, składa się stała opłata za korzystanie z usług dostawcy oraz kwota wyliczona, na podstawie zużytych zasobów przez aplikacje. Przykładem platformy jako usługi może być Google Aps Engine czy Microsoft Azure. Środowisko Microsoft oferuje na tym polu odrobine więcej niż Google, jednakże jest trudniejsze w konfiguracji i obsłudze. Na platformie Microsoft Azure można uruchomić dowolną aplikację, napisaną w języku z rodziny .Net, Javie, PHP czy C.

### Infrastruktura jako usługa (IaaS)

W ramach IaaS klientowi końcowemu, oferowana jest cała niezbędna infrastruktura informatyczna, potrzebna do stworzenia chmury obliczeniowej [14]. Dostawca gwarantuje dostarczyć:

* sprzęt: serwery, pamięci masowe
* oprogramowanie: wirtualizacja, hipernadzorca
* serwis: wymiana/naprawa sprzętu w przypadku awarii
* czynniki niezbędne do poprawnego działania sprzętu: zasilanie, klimatyzacja

Klient, ma możliwość utworzenia maszyn wirtualnych, podając ich parametry czy system operacyjny. Ma szerokie możliwości kontroli i konfiguracji. Za pomocą konsoli menadżera sprzętu, może tworzyć kopie zapasowe, uruchamiać maszyny wirtualne. Użytkownik płaci za ilość zasobów, a nie za sprzęt. Eliminuje to ogromne koszty związane z zakupem sprzętu potrzebnego do budowy infrastruktury. Klient jedynie „wypożycza” sprzed od dostawcy, nie ma wiedzy o tym, co się dzieje ze sprzętem, który jest mu udostępniony. Hardware, może być wymieniany, dublowany czy rozpraszany, a klienta będą interesowały tylko dostępny zasoby.

Płatności za chmury typu IasS nie są skomplikowane. Zazwyczaj dokonuje się opłaty za czas działania wynajmowanej maszyny. Czynnikiem mającym wpływ na opłatę, są również parametry tej maszyny. Firma Amazon zdaję się być liderem na rynku chmur obliczeniowych opartych na modelu IaaS. Ma w swojej ofercie ciekawą usługę zwaną Elastic Compute Cloud, opartą na technologii wirtualizacji Xen. Amazon, oferuje również ciekawe dodatkowe usługi jak np.: Elastic Load Balancing, który automatycznie równoważy obciążenie pomiędzy wirtualnymi serwerami.

# Zastosowanie chmury obliczeniowej

## Szybsze i łatwiejsze wdrażanie rozwiązań

Chmura obliczeniowa i rozwiązania tupu PaaS i IaaS niosą duże korzyści dla potencjalnego użytkownika. Dla przedsiębiorcy, oznaczają one mniejszą barierę wejścia w nowe projekty informatyczne oraz szybsze wprowadzenie produktów na rynek. Firma, nie ponosi dużych kosztów inwestycyjnych związanych z zakupem sprzętu, licencji czy oprogramowania, już na samym starcie danego przedsięwzięcia informatycznego. W przypadku niepowodzenia w danym projekcie, wystarczy zrezygnować z usług danego dostawcy, nie trzeba martwić się o zakupiony wcześniej sprzęt. Straty w takim scenariuszu nie będą, aż tak duże.

## Środowiska testowe i developerskie

Prawdopodobnie jeden, z lepszych scenariuszy, w którym można wykorzystać potencjał chmury. Utworzenie środowiska testowego lub developerskiego również, wiązało by się z zakupem sprzętu i organizacją infrastruktury. Następnie infrastrukturę należało by odpowiednio skonfigurować pod dane środowisko lub środowiska testowe. Takie zabiegi są bardzo kosztowne, mogą wydłużyć fazę testowania, a tym samym cały projekt. Chmura, może skrócić czas utworzenia środowiska oraz zminimalizować jego koszty. Dodatkowo wszystko można łatwo dostosowywać, konfigurować do swoich potrzeb i wymagań.

## Analiza danych (ang. Big Data Analytics)

Jednym z aspektów oferowanych przez wykorzystanie technologii w chmurze, jest możliwość dotarcia do ogromnych ilości zarówno danych strukturalnych, jak i niestrukturalnych, które wykorzystują korzyści związane z uzyskaną wartością biznesową [15].

Sprzedawcy oraz dostawcy obecnie pozyskują informacje pochodzące ze statystyk sprzedażowych i kierują kampanie reklamowe i marketingowe do konkretnego segmentu populacji. Platformy sieci społecznościowych, są obecnie podstawą do analizy wzorców zachowań, które organizacje wykorzystują w celu uzyskania znaczących informacji.

## Przechowywanie plików

Podstawowym oczywiście zastosowaniem chmury jest przechowywanie i udostępnianie dużej ilości danych. Oferuje ona przechowywanie plików i ich pobieranie przez dowolny interfejs internetowy. Interfejsy usług internetowych są zwykle proste. W każdej chwili i w każdym miejscu mamy wysoką dostępność, szybkość, skalowalność i bezpieczeństwo dla środowiska. W takiej sytuacji firmy płacą tylko za ilość pamięci masowej, którą rzeczywiście konsumują, i robią to bez obaw o nadzorowanie codziennej konserwacji sprzętu (np.: pamięci masowej). Świętnym rozwiązaniem, jest stosowanie chmury do przechowywania kopi zapasowych. W tym scenariuszu, każda stacja robocza miałaby łatwy i szybki dostęp i w prosty sposób, można by zaplanować wykonanie przywracania kopi zapasowej na wielu stacjach jednocześnie.

## Aplikacje, multimedia i rozrywka

W Internecie można spotkać „chmurowe” odpowiedniki popularnych aplikacji desktopowych. Pierwszym przykładem jest skype, który ostatnio został przeniesiony właśnie do chmury. YouTube udostępnia usługę przechowywania swoich multimediów na dysku chmurowym.

Od dłuższego czasu dostępne są pakiety biurowe (edytor tekstu, arkusz kalkulacyjny, edytor prezentacji) od Microsoft’u i Google’a w wersjach internetowych, dostępnych przez przeglądarkę internetową. Użytkownicy często narzekają na zbyt wolne i obciążające działanie programów antywirusowych. Część producentów programów antywirusowych znalazła rozwiązanie tego problemu oferując wersje swoich produktów działających w chmurze np.: Panda Cloud Antivirus. Rynek gier również, pozytywnie zareagował na możliwości jakie stwarza cloud computing. Wizja, w której najnowsze, najbardziej wymagające gry, są uruchamiane i renderowane na wysokiej jakości zdalnym serwerze, a tylko wyrenderowane klatki, trafiają za pośrednictwem sieci do odbiorcy, wydaję się być bardzo atrakcyjna [16]. Wyeliminować można w ten sposób, chociażby aspekty takie jak: niespełnienie wymagać przez własny komputer czy platforma operacyjna na której nie uruchomiłby się dany tytuł.

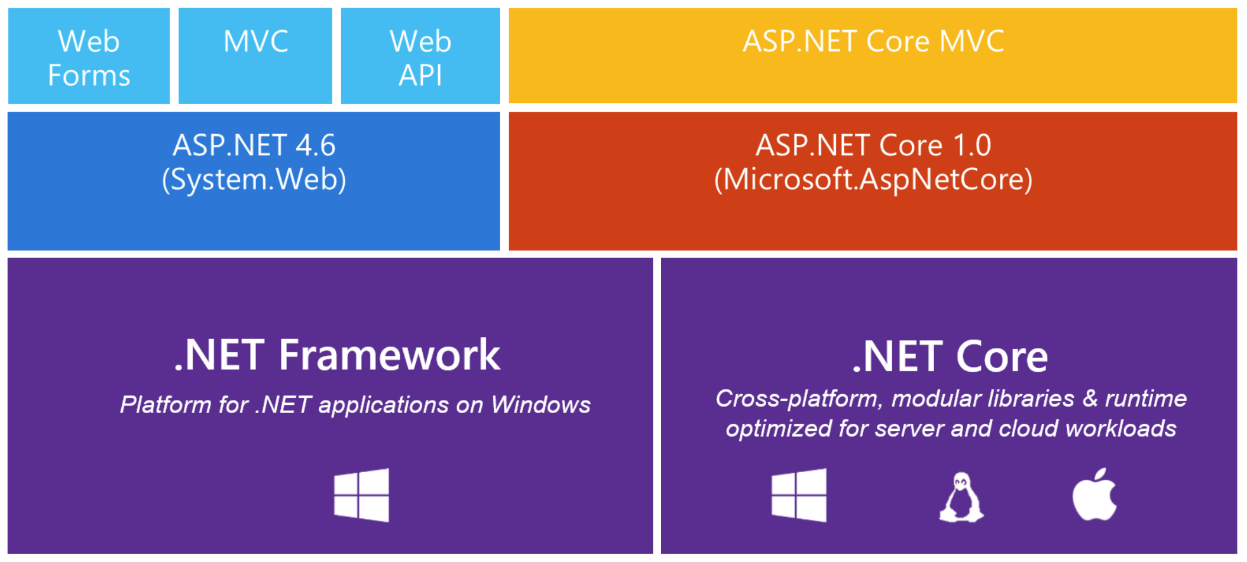
# Wykorzystane narzędzia i technologie

W pracy magisterskie wykorzystany szeroki zakres narzędzi, framework’ów i bibliotek. Najważniejszym jes,t opracowany przez Microsoft, framework do aplikacji webowych ASP .NET Core. Językiem programowania, w którym napisano aplikacje jest c#.

## Technologie

### Asp .NET Core

Do roku 2016 roku Microsoft był dość sceptycznie nastawiony na wieloplatformowość. Wszystkie .NET-owe biblioteki były ograniczone tylko do jednej platformy – Windows’a. Jednakże, w 2016 roku nastąpił przełom i największe przedsiębiorstwo w branży informatycznej postanowiło otworzyć się na inne platformy i wypuścić na rynek produkt, który będzie open-source (otwarty kod źródłowy) oraz wieloplatformowy (Rysunek 5.1).



Rysunek 5.2 Różnice nowego środowiska .NET Core w porównanie z poprzednią wersją

[http://thienn.com/images/aspnetcore.png dostęp z dnia 11.062017](http://thienn.com/images/aspnetcore.png%20dostęp%20z%20dnia%2011.062017)

Tak powstał, całkiem nowy framework, zbudowany od podstaw - .NET Core. Zdecydowano, aby wykonać aplikacje webową w ASP .NET Core, ze względu na jego przyszłościowy charakter oraz na łatwość konfiguracji. Dodatkowo, w tej wersji framework’a wprowadzono liczne zmiany, które przyczynią się to efektywniejszej i wydajniejszej pracy m.in.: cały framework jest tak naprawdę zestawem komponentów, które można swobodnie dodawać i usuwać, lepsza struktura projektu, czy wbudowana obsługa wstrzykiwania zależności [17].

### Asp Identity 2

Każdy system, aplikacja lub prosta strona internetowa typu blog, wymaga zaimplementowania mechanizmu logowania czy wprowadzenia różnych ról dla użytkowników. Ręczne tworzenie wszystkich komponentów odpowiedzialnych za obsługę logowania, przechowywania i hashowania haseł, było by zbyt czasochłonne, jak i również nieodpowiednie zaimplementowanie tak ważnego komponentu aplikacji naraziło by ją na nieautoryzowany dostęp. Dlatego, w serwisie pośrednictwa pracy zdecydowano wykorzystać bibliotekę ASP Identity 2, która zapewnia wszystkie niezbędne elementy, które cechuje prosta konfiguracja i bezpieczeństwo stojące na wysokim poziomie [18].

### Entity Framework Core 1.0

Do komunikacji i wymiany danych z bazą danych, wykorzystano narzędzie ORM (ang. Object-Relational Mapping). W mniejszych i średnich projektach idealnie znajduje swoje zastosowanie, nie wymaga znajomości ani jezyka zapytań SQL ani narzędzi do administrowania baza danych, działania na kontekście bazy danych są prawie takie same jak na zwykłych listach czy tablicach, dlatego, że jest napisany na interfejs IQuerryable, odpowiednik interfejsu IEnumerable. [Entity Framework Core](https://docs.microsoft.com/en-us/ef/core/) to lżejsza wersja starszego brata [Entity Framework](http://msdn.com/data/ef). Jego najważniejsza zaleta to możliwość wykorzystania w projektach .NET Core, czyli tym samym użycie na wielu platformach [19].

### Chmura obliczeniowa Microsoft Azure

Kwestie chmury zostały już w pracy magisterskiej szczegółowo omówione i nie ma potrzeby, aby powtarzać informacje dotyczące chmury. Należy nadmienić tylko, że wybraną chmure Azure, ze względu na jej duże możliwości, jak i również darmowe licencje DreamSpark dla studentów. Zarządzanie zasobami odbywa się z poziomi samego Visual Studio Enterprise 2017, sporadycznie za pomocą internetowego portalu portal.azure.com.

### Bootstrap

Szablon portalu pośrednictwa pracy został wykonany z pomocą HTML, CSS oraz framework’a Bootstrap, który udostępnia szereg klas nadających ciekawy wygląd wielu komponentów na stronie www. Uzyskany został, dzięki tej bibliotece również efekt responsywności, co ma szczególne znaczenie dla użytkowników logujących się do serwisu za pomocą mobilnych urządzeń

## jQuerry

Niektóre fragmenty widoków lub ich zachowania zostały napisane w JavaScript’cie z użyciem biblioteki jQuerry.

## Narzędzie

### Microsoft Visual Studio 2017

Głównym narzędziem, w którym napisano aplikację jest Microsoft Visual 2017 w wersji Enterprise. Dzięki uczelnianemu programowi DreamSpark, była możliwość pobrania tej wersji programu Visual Studio wraz z kluczem licencyjnym. Visual Studio oferuje szerokie możliwości instalowania dodatkowych bibliotek czy wtyczek, administracji serwerów i baz danych, debugowania i testowania kodu [20]. Skorzystano również dodatkowej biblioteki do refaktoryzacji kodu, ReSharper’a. Dodatek pobrano z oficjalnej strony jetbrains.com, na licencji studenckiej.

# Projekt Aplikacji

Proces tworzenia aplikacji webowej podzielono na kilka etapów. Następnie, starano się etapy rozbijać na mniejsze pod zadania i je realizować. Niektóre etapy można było zrównoleglić, inne natomiast musiały zostać wykonane priorytetowo ze względu na blokowanie zadań w innych etapach.

Przebieg projektowania i implementacji serwisu sprowadzał się do kilku głównych części:

1. Pomysł – nazwa serwisu, funkcjonalności, założenia, określenie grupy docelowych odbiorców.
2. Wybór technologii i narzędzi
3. Przygotowanie szablonów w HTML’u opartych na Bootstrapie
4. Baza danych – rozplanowanie encji
5. Architektura serwisu
6. Konfiguracja projektu - instalacja i konfiguracja niezbędnych dodatków, bibliotek i wtyczek
7. Implementacja funkcjonalności
8. Refaktoryzacja kodu
9. Deployment aplikacji
10. Testy

## Koncepcja serwisu

Serwis wykonany w ramach pracy magisterskiej, jest przeznaczony dla grupy programistów poszukujących pracy, jak i pracowników, którzy poszukują specjalistów z branży IT. Serwis zyskał nazwę Team Leasing , wolne tłumaczenie tego zwrotu to wydzierżawienie zespołu (ang. to lease – wydzierżawić), idea takiej, a nie innej nazwy pochodzi od jej funkcjonalności, wyróżniającej cały serwis na tle innych tego typu portali z ogłoszeniami. Serwis umożliwia utworzenie oferty pracy, przeznaczonej dla całego zespołu programistów, a nie jednego developera.

## Główne założenia aplikacji

Główne przeznaczenie serwisu to umożliwienie pracodawcom znalezienie dobrego specjalisty, a developerom – pracy.

Funkcjonalności serwisu podzielono na dwie grupy, a w zasadzie na trzy, bo oprócz tych skierowany dla użytkowników, którzy zarejestrują się jako programiści i tych którzy zarejestrują się w serwisie jako pracodawcy, należy wyróżnić anonimowych gości w serwisie którzy jeszcze nie posiadają konta w bazie danych.

Część z funkcjonalności oczywiście będzie podobna w tych grupach.

Anonimowy użytkownik ma możliwość:

* Rejestracji w serwisie jako programista lub pracodawca
* Przeglądanie/wyszukiwanie ofert pracy bądź profilów programistów
* Przeglądanie/wyszukiwanie zleceń dla całego zespołu programistów

Użytkownik, który posiada w systemie role „Programista”

* Edycje własnego konta: zmiana danych np.: dane personalne, dodanie pliku zawierającego CV czy pliku ze zdjęciem
* Aplikowanie na wybrane oferty pracy
* Możliwość negocjacji warunków danej oferty

Użytkownik, który posiada w systemie role „Pracodawca”

* Edycja własnego konta
* Przeglądanie profilów programistów i ich wrzuconych plików
* Wysyłanie propozycji pracy programistom
* Tworzenie ogłoszeń - ofert pracy
* Tworzenie projektów – zleceń dla całego zespołu programistów
* Negocjowanie warunków zatrudnienia

## Struktura bazy danych

W warstwie danych serwisu, utworzono barze danych MSSQL do przechowywania wszystkich informacji o użytkownikach. Do utworzenia diagramów bazy danych, wykorzystano narzędzie Miscrosoft SQL Server Managment Studio. To środowisko, oferuje duże możliwości: tworzenie baz danych, zarządzanie nimi, generowanie skryptów, importowanie baz danych czy też generowanie diagramów z istniejących już baz danych. Na *Rysunek 6.4* oraz *Rysunek 6.5* przedstawiono pełne schematy bazy danych, wygenerowane za pomocą wbudowanej funkcji w środowisku Microsoft Managment Studio. Najistotniejszy z punktu widzenia samej logiki aplikacji, jest diagram przedstawiony na *Rysunek 6.4*. Założono, że serwis będzie dotyczył dwóch grup użytkowników, programistów i pracodawców, na tej podstawie utworzono dla nich encje – DeveloperUser oraz EmplyeeUser - przechowywujące dane personalne użytkowników (Rysunek 6.1).



Rysunek 6.1 Tabele dla użytkowników DeveloperUser oraz EmplyeeUser

Aplikacja oferuje możliwości tworzenia ofert pracy, tworzenia projektów dla całych team’ów developerów, wysyłanie określonych propozycji dla wybranych specjalistów, oraz negocjowanie warunków oferty pomiędzy programistą, a potencjalnym pracodawcą. Logika przetwarzająca i realizująca zadania w ramach tej funkcjonalności, zapisuje dane w tabelach Job (oferta pracy), Offer (propozycja pracy), Negotiation (negocjacje) oraz Project (projektu dla całego zespołu programistów)(*Rysunek 6.2*)



Rysunek 6.2 Tabele Project, Offer, Negotation oraz Job

Dodatkowo przewidziano encje dla wiadomości wysłanych z podstrony kontaktowej, oraz statyczną tabele przechowywującą informacje o technologiach/językach/frameworkach, które można w aplikacji wybrać, przy określaniu chociażby kryteriów wyszukiwania interesujących ofert pracy (Rysunek 6.3).



Rysunek 6.3 Tabele Message oraz Technology

Relacje w bazię danych wyglądają następująco:

* Programiści mogą aplikować na wiele stanowisk (relacja wiele do wielu)
* Pracodawca może mieć wiele ofert pracy (relacja jeden do wielu)
* Pracodawca może mieć wiele projektów (relacja jeden do wielu)
* Pracodawcy mogą składać własne propozycje pracy dla wybranych programistów

( relacja wiele do wielu)

* Dana propozycja pracy jest negocjowana w ramach pojedynczej wiadomości (jeden do jednego)

Podczas projektowania serwisu wykonanego w ramach pracy magisterskiej, postanowiono wykorzystać gotową bibliotekę zawierającą wszystkie niezbędne elementy odpowiadające za bezpieczeństwo, hashowanie haseł, czy autoryzacje użytkowników. Bezpieczeństwo osób korzystających z platformy jest priorytetem, dlatego nie implementowano własnych rozwiązań, a zdecydowano się wykorzystać istniejące i sprawdzone rozwiązania. To wszystko zapewnia biblioteka open source – ASP IDENTITY 2. Dostarcza encje (*Rysunek 6.5*), które przechowywujący wrażliwe dane użytkownika, role użytkowników czy tokeny. Zapewnia mechanizmy hashownaia, tokenowania oraz autoryzacje na najwyższym standardzie - Client Certificate Authetication (uwierzytelnianie w oparciu o certyfikat X.509 klienta)[22]



Rysunek 6.4 Schemat tabel bazy danych

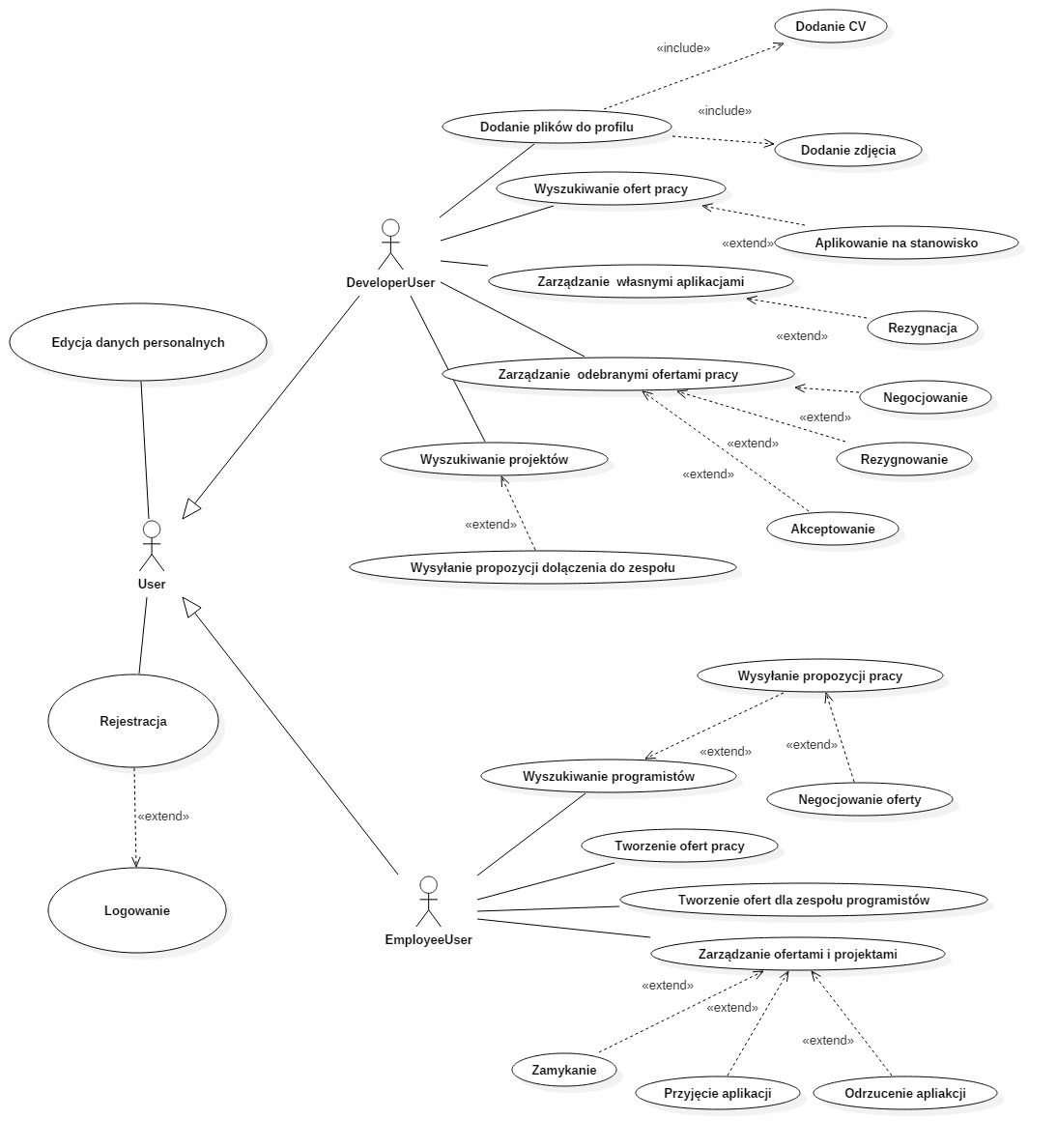


Rysunek 6.5 Tabele biblioteki ASP IDENTITY 2

## Ogólny opis projektu

### Diagram przypadków użycia

Diagram przypadków użycia (*Rysunek 6.6*), schematycznie przedstawia funkcje aplikacji, w zależności rodzaju użytkownika.

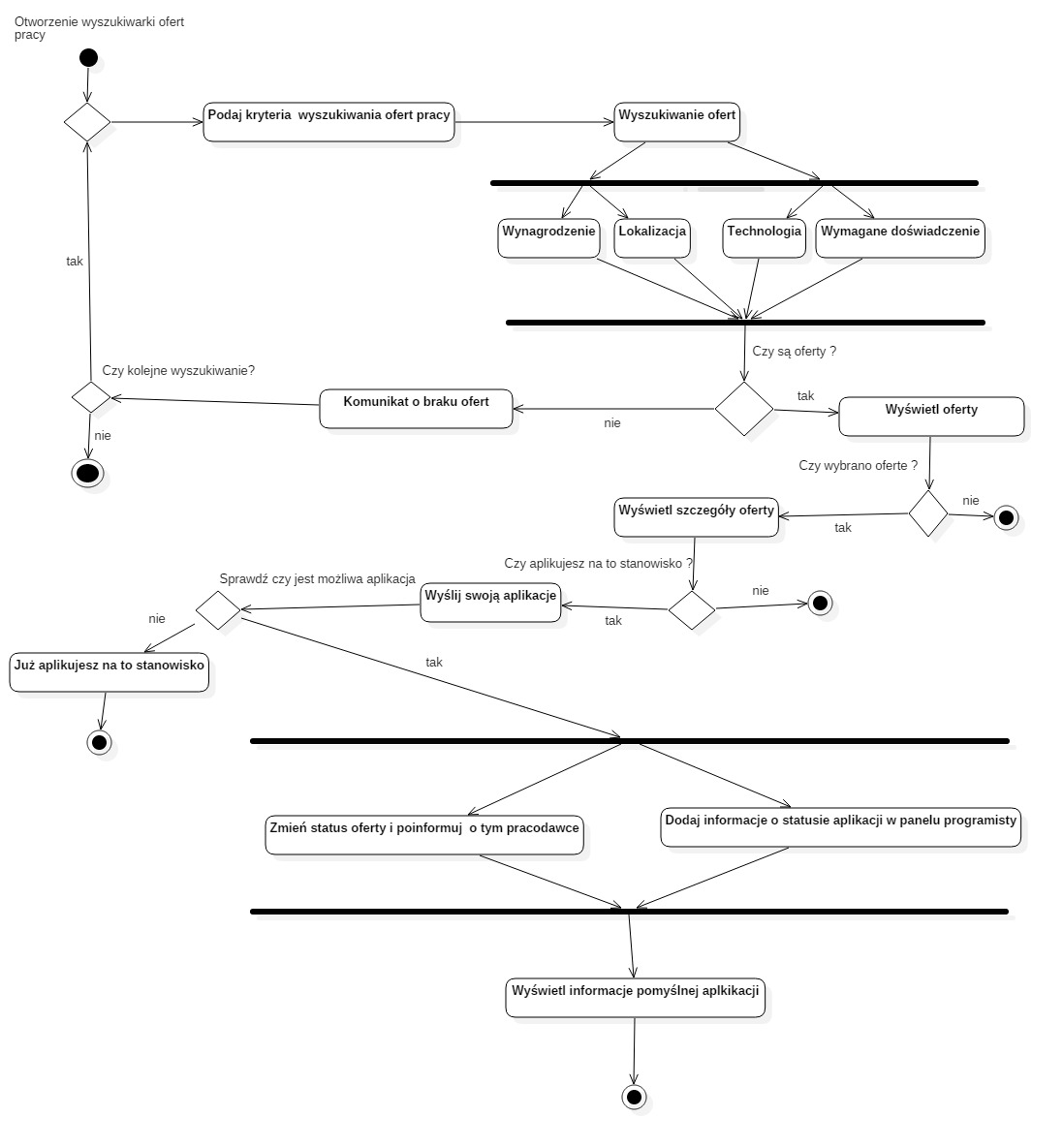


Rysunek 6.6 Diagram przypadków użycia

## Diagramy aktywności

### Wyszukiwanie oraz aplikacja na daną ofertę pracy

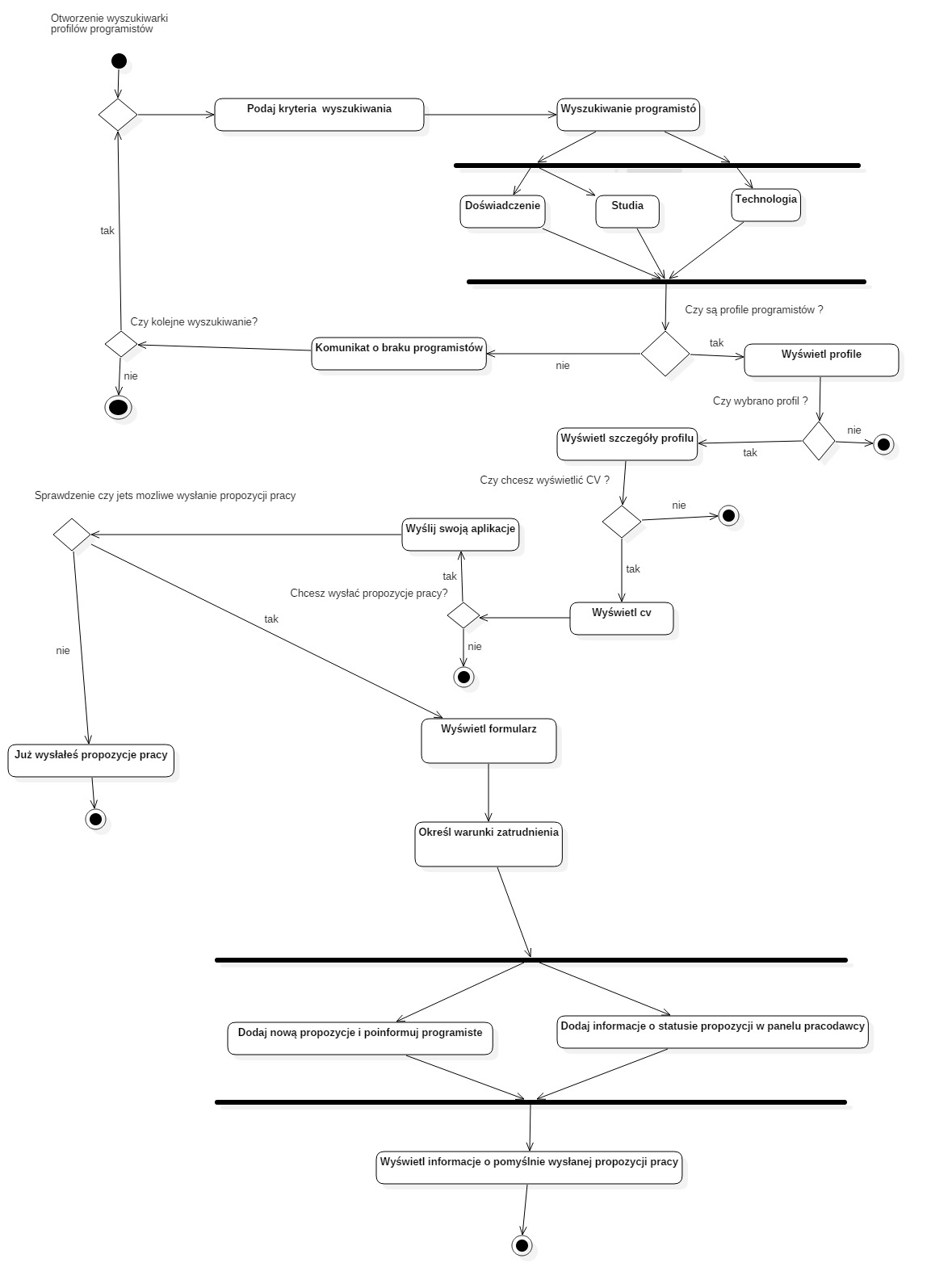
Na *Rysunek 6.7* przedstawiono diagram aktywności, który dotyczy wyszukiwania ofert pracy bądź projektów oraz składania aplikacji na wybraną ofertę pracy.



Rysunek 6.7 Diagram aktywności wyszukiwania oraz składania aplikacji na daną ofertę pracy

### Wyszukiwanie profilów programistów oraz składanie propozycji pracy

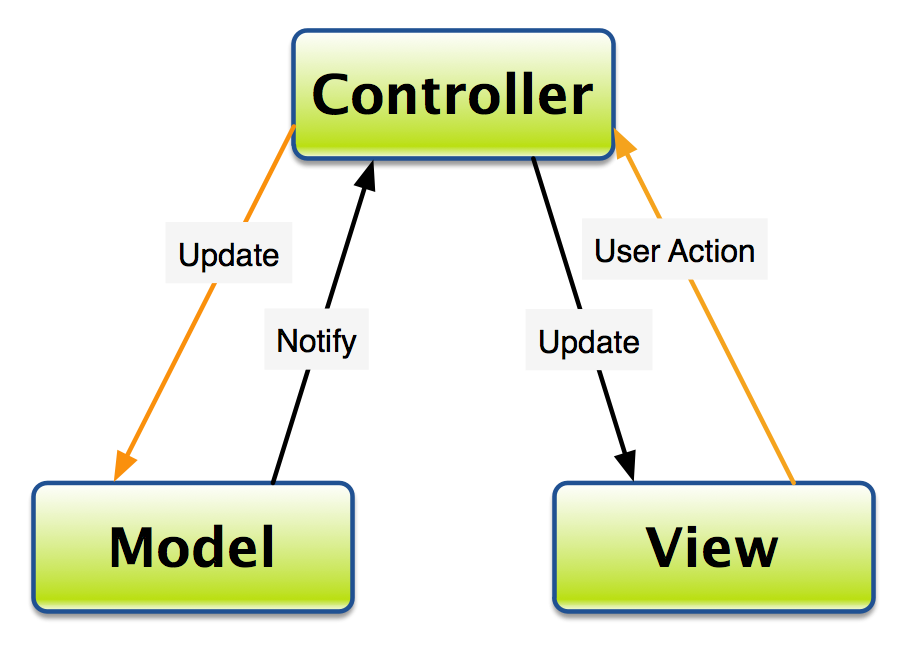
Na *Rysunek 6.8* przedstawiono diagram aktywności wyszukiwania profilów programistów i wysyłania im ofert pracy.



Rysunek 6.8 Diagram aktywności wyszukiwania programistów i wysyłania im ofert pracy

## Architektura systemu pośrednictwa pracy

Serwis wykonano zgodnie ze wzorcem MVC stosowanym w większości nowoczesnych aplikacji, gdzie następuje podział na trzy warstwy: warstwę widoków, warstwę kontrolerów i modeli. Separacja poszczególnych warstw jest sytuacją pożądaną i niesie ze sobą wiele korzyści. Dodatkowo warstwy te ze sobą luźno powiązane, co jest zgodne z aktualnymi standardami tworzenia oprogramowania. Dzięki takie strukturze, można tworzyć serwisy, które są skalowane i łatwe w utrzymaniu i rozwoju [21]. W warstwie widoku, znajdują się tylko podstrony zakodowane w html’u wraz z wygenerowanym kodem silnika Razor, dzięki, któremu można po stronie widoków generować za pomocą składni języka c# - kod html’a. Warstwa kontrolerów jest pewnego rodzaju nośnikiem i przekaźnikiem informacji, umożliwia komunikacje pomiędzy warstwą widoków i modeli. Natomiast warstwa modeli zarządza danymi. Ogólny przepływ danych pomiędzy warstwami jest przedstawiony na *Rysunek 6.9*



Rysunek 6.9 Wzorzec MVC

Użytkownik z poziomu warstwy widoku może zmienić dane, które następnie trafiają do kontrolera, a ten już bezpośrednio może wykonać działania na warstwie danych. Wszystkie operacje na bazie danych są wykonywane za pośrednictwem warstwy kontrolerów.

Dodatkowo systemie informatycznym można zaprojektować dodatkowe warstwy realizujące określone operacje. Można utworzyć warstwę logiki biznesowej, w której działają serwisy lub inne komponenty zajmujące się logicznym przetwarzaniem danych, czy warstwę repozytorium danych.

# Implementacja

## Logowanie

Mechanizmy logowania oraz rejestracji udostępnia biblioteka ASP .NET Core Identity.

Najistotniejszą kwestią jest dodanie mechanizmów logowania i rejestracji w Service (interfejs IServiceColletion) (Listing 7.1). Określa się tu również kryteria jakie będą obowiązywała podczas rejestracji np.: długość, hasła, obecność dużych małych liter w haśle, czy adres email ma być unikalny. Dodatkowo ustawiono, aby w wersji produkcyjnej aplikacja używała do uwierzytelniania protokołu HTTPS, w wersji developerskiej, nie jest on aż tak wymagany.

Widok logowania użytkowników jest dość prosty, należy podać nazwę użytkownika oraz hasło (Rysunek 7.1);

Listing 7.1 Mechanizm autoryzacji i jego ustawienia

services.AddIdentity<User, IdentityRole>(c =>

{

c.Password.RequiredLength = 4;

c.User.RequireUniqueEmail = true;

c.Lockout.MaxFailedAccessAttempts = 5;

c.Cookies.ApplicationCookie.LoginPath = "/login/login";

})

.AddEntityFrameworkStores<TeamLeasingContext>()

.AddDefaultTokenProviders()

.AddUserManager<OptimizedDbManager>();

services.AddMvc(config =>

{

if (\_env.IsProduction())

config.Filters.Add(new RequireHttpsAttribute());

});

services.AddSession();

services.AddLogging();

Metoda, która obsługuje logowanie znajduję się w kontrolerze Login i nie jest skomplikowana (Listing 7.2). W parametrach pobiera LoginViewModel, który zawiera nazwę użytkownika oraz hasło, a następnie za pomocą SignInManagera – wbudowanego w biblioteke Identity – próbuje dokonać logowania, w przypadku błędnego zalogowania użytkownik otrzyma stosowne komunikaty.

Listing 7.2 Metoda obsługująca logowanie

[HttpPost]

public async Task<IActionResult> Login(LoginViewModel vm)

{

if (ModelState.IsValid){

var result = await \_signInManager.PasswordSignInAsync(vm.Username, vm.Pasword, false, false);

if (result.Succeeded)

{

if (string.IsNullOrWhiteSpace(vm.ReturnUrl))

{

return RedirectToAction("Index", "Home");

}

else

{

return Redirect(vm.ReturnUrl);

}

}

else

{

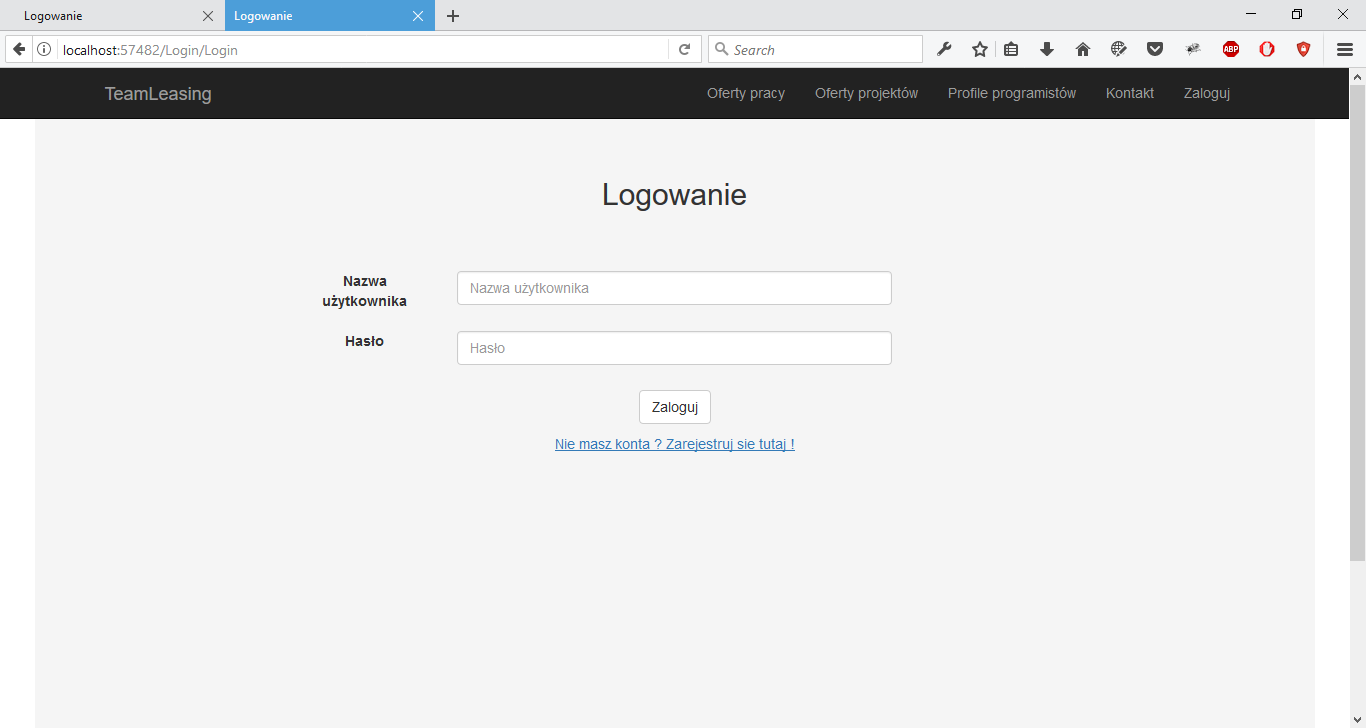
ModelState.AddModelError("", "Nieprawidłowy login lub hasło !");

}

}

return View("Login");

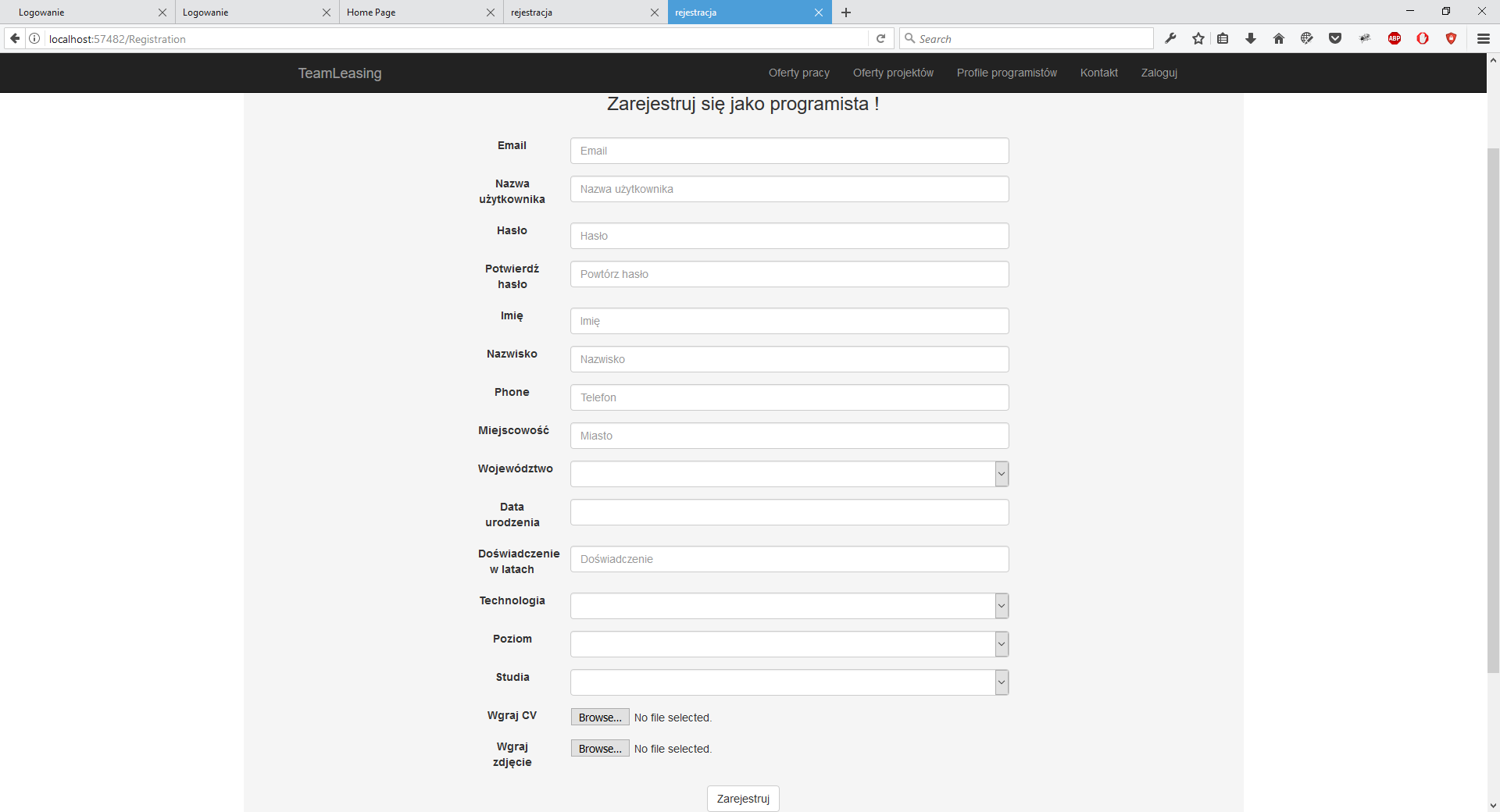
}



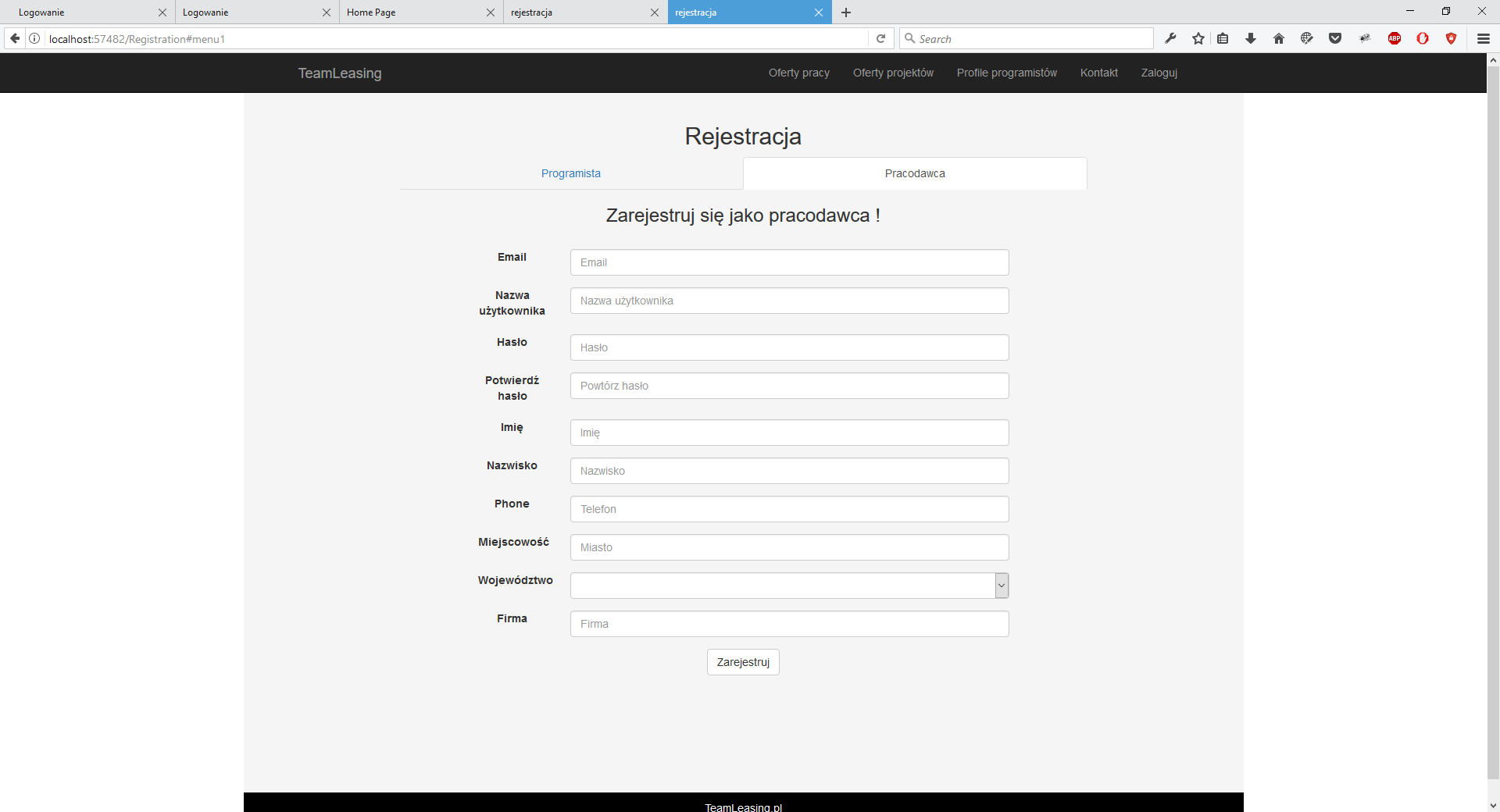
Rysunek 7.1 Ekran logowania

## Rejestracja

Obsługa rejestracji użytkownika jest już bardziej skomplikowana, chociażby ze względu na bardziej rozbudowany formularz. Rejestrując się jako pracodawca (*Rysunek 7.3 Rejestracja pracodawcy*) wystarczy wypełnić podstawowe dane personalne, takie jak imię, nazwisko, telefon czy nazwę firmy. Natomiast rejestracyjny formularz dla programisty zawiera więcej pól (*Rysunek 7.2*). Użytkownik może określić technologie w której się specjalizuje czy swój poziom umiejętności (junior, regular czy senior developer). Dodatkowo ma możliwość dodania zdjęcia czy CV.



Rysunek 7.2 Rejestracja programisty



Rysunek 7.3 Rejestracja pracodawcy

Rejestracja użytkownika odbywa się z pomocą UserManagera, który jest wbudowany w bibliotekę Identity i udostępnia metody do tworzenia , sprawdzania informacji czy modyfikowania użytkowników. Metoda CreateAsync tworzy użytkownika, następnie meto AddRole nadaje mu odpowiednią role w serwisie, a następnie loguje użytkownika do serwisu

W przypadku błędnego wprowadzenia danych, użytkownik otrzyma komunikaty informujące go o błędnej walidacji.

Listing 7.3 Metody do rejestracji developera oraz pracodawcy

private async Task<IActionResult> CreateUser(RegistrationDeveloperViewModel vm)

{

User user = new User();

user = \_mapper.Map<User>(vm);

var result = await \_manager.CreateAsync(user, vm.Password);

if (result.Succeeded)

{

var newUser = \_teamLeasingContext.Users.Find(user.Id);

DeveloperUser developerUser = await CreateDeveloperUser(vm, newUser);

await \_teamLeasingContext.DeveloperUsers.AddAsync(developerUser);

await \_teamLeasingContext.SaveChangesAsync();

await \_manager.AddToRoleAsync(\_teamLeasingContext.Users.Find(user.Id), Roles.Developer.ToString());

await \_signInManager.SignInAsync(\_teamLeasingContext.Users.Find(user.Id), true, null);

return RedirectToAction("Index", "Home");

}

throw new Exception(message: "Utworzenie użytkownka z przyczyn niewyjaśnionych nie powiodło się");

}

private async Task<DeveloperUser> CreateDeveloperUser(RegistrationDeveloperViewModel vm, User user)

{

DeveloperUser developerUser = new DeveloperUser();

developerUser = \_mapper.Map<DeveloperUser>(vm);

developerUser.Technology = \_teamLeasingContext.Technologies

.Where(t => t.Name.ToLower() == vm.ChoosenTechnology.ToLower())

.ToList()

.FirstOrDefault();

developerUser.UserId = user.Id;

if (vm.CvFile!=null)

{

developerUser.Cv = await \_userService.UploadService.UploadCvFile(vm.Name, vm.Surname, vm.CvFile);

}

if (vm.PhotoFile!=null)

{

developerUser.Photo = await \_userService.UploadService.UploadPhotoFile(vm.Name, vm.Surname, vm.PhotoFile);

}

return developerUser;

}

## Edycja konta

W serwisie istnieje możliwość prostej edycji własnego konta (*Rysunek 7.4*), zarówno programista jak i pracodawcą mogą zmienić niektóre dane dotyczące ich konta. Można edytować wybrane elementy, użytkownik może zmienić swoje zdjęcie, może je całkowicie usunąć, może również wgrać nowsze wersje swojego CV. Edycja konto w zasadzie nie różni się zbytnio od tworzenia nowego użytkownika, róznica sprowadza się jedynie to wywołania meto Update zamiast metody Create. Natomiast dodanie plików przez użytkownika na serwer wymaga już większej uwagi, Listing 7.1 przedstawia fragment kodu odpowiedzialą za tę funkcjonalność. Najważniejsze, to określić odpowiednią ścieżkę do folderu, w którym będą przechowywane zdjęcia, dzięki interfejsowi IHostingEnvironment można w łatwy sposób wywołać metodę WebRootPath i określić ścieżkę zapisu pliku i zapisać ten plik na serwerze.

Listing 7.4 Dodawanie plików na serwer

public async Task<string> UploadPhotoFile(string name, string surname, IFormFile photoFile, bool isFileExist=false)

{

var pathToFile = $"/UploadFile/Photo/{surname.Trim()}\_{name.Trim()}.jpg";

var uploadPhotoPath = \_environment.WebRootPath + pathToFile;

CheckIsFileExist(isFileExist, uploadPhotoPath);

await CopyFile(photoFile, uploadPhotoPath);

return pathToFile;

}

public async Task<string> UploadCvFile(string name, string surname, IFormFile cvFile, bool isFileExist=false)

{

var pathToFile = $"/UploadFile/Cv/{surname.Trim()}\_{name.Trim()}.pdf";

var uploadCvPath = \_environment.WebRootPath + pathToFile;

CheckIsFileExist(isFileExist, uploadCvPath);

await CopyFile(cvFile, uploadCvPath);

return pathToFile;

}

private void CheckIsFileExist(bool isFileExist, string pathToFile)

{

if (isFileExist)

{

System.IO.File.Delete(pathToFile);

}

}

private async Task CopyFile(IFormFile file, string path)

{

if (file != null)

{

using (var fileStream = new FileStream(path, FileMode.Create))

{

await file.CopyToAsync(fileStream);

}

}

else

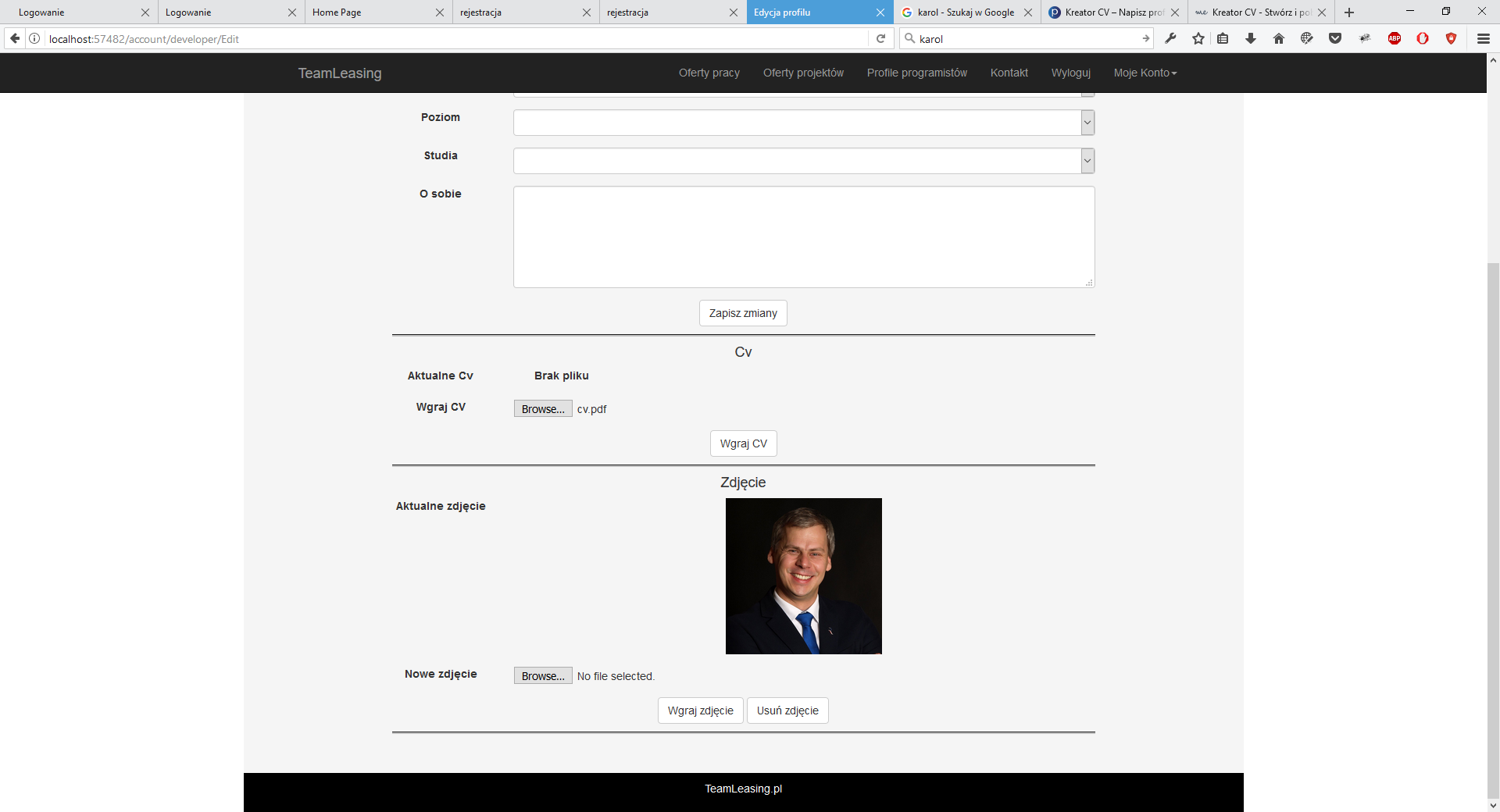
{

throw new Exception(message: "Bład wgrywania pliku");

}

}





Rysunek 7.4 Widok edycji konto

## Wyszukiwanie ofert/projektów/profilów programistów

### Wyszukiwarka

Wyszukiwanie ofert lub też większych projektów jest bardzo proste. Wystarczy przeglądać całą dostępną ofertę, lub też podać szczegółowe kryteria i przefiltrować wyniki. Użytkownik ma możliwość określenia takich parametrów jak: wynagrodzenie, lokalizacja, technologie czy wymagany poziom umiejętności na daną ofertę pracy (*Rysunek 7.5*). Kod odpowiedzialny za filtrowanie wyników znajduję się na Listing 7.1. W metodzie GetSearchResult sprawdzane są kryteria, te podane przez użytkownika z tymi, które zawierają wszystkie oferty pracy dostępne w bazie danych. Dzięki mechanizmowi LINQ i metodom extensions (where, select oraz contain), w prosty sposób następuje sprawdzenie czy warunek szukania jest prawdą czy fałszem i które wyniki spełniają podane kryteria. Dodatkowo, wydzielono do osobnej klasy kod nazwany Helper’em (*Listing 7.6*), który jest uniwersalny i wykorzystywany jest również przy wyszukiwaniu programistów czy projektów. W przypadku wyszukiwania projektów czy profilów programistów, zasady wyszukiwania i wykorzystane metody są bardzo podobne, nie ma potrzeba ich dokładnie omawiać.

Listing 7.5 Metoda filtrująca oferty pracy

private async Task<List<Job>> GetSearchResult(SidebarEmployeeViewModel vm,

List<Job> jobs)

{

return await Task.Run(async () =>

{

var tech = \_searchHelper.Aplly<Job>(w => vm.TechnologyNameValuePairs

.Where(s => s.Value)

.Select(s => s.Name)

.Contains(w.Technology.Name), jobs);

var level = \_searchHelper.Aplly<Job>(w => vm.LevelNameValuePairs

.Where(s => s.Value)

.Select(s => s.Name)

.Contains(w.Level), jobs);

var experience = \_searchHelper.Aplly<Job>(w => w.Price > vm.SalaryMin

&& w.Price < vm.SalaryMax, jobs);

var searchingResult = await \_searchHelper.Intersection<Job>(tech, level, experience);

return searchingResult.ToList();

});

Listing 7.6 Helper – metody pomocnicze

public Task<IEnumerable<T>> Intersection<T>(params IEnumerable<T>[] paramsList)

{

return Task.Run(() =>

{

paramsList = paramsList.Where(w => w.Count() != 0).ToArray();

for (int i = 1; i < paramsList.Length; i++)

{

paramsList[0] = paramsList[0].Intersect(paramsList[i]).ToList();

}

return paramsList[0];

});

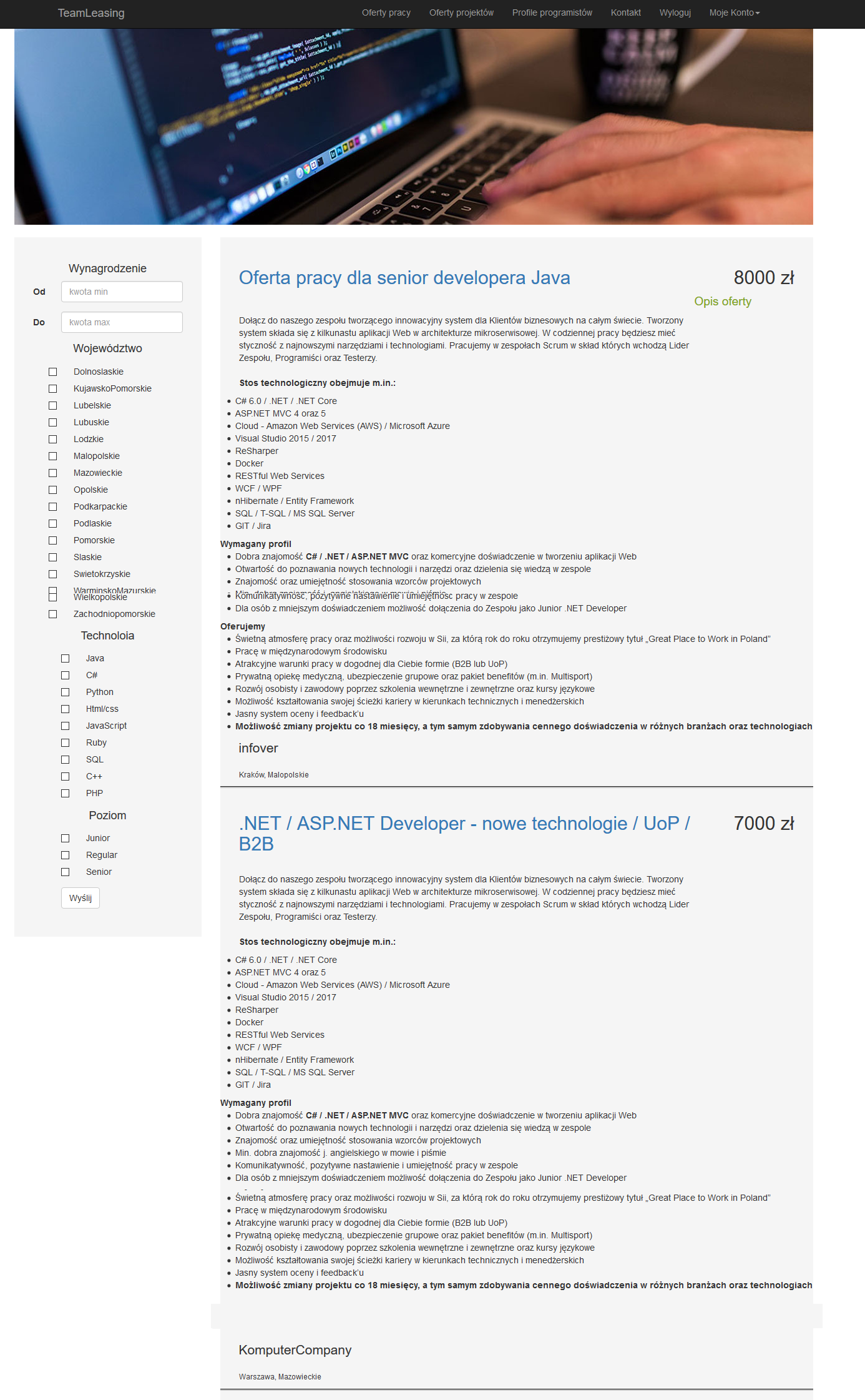
}

public List<T> Aplly<T>(Func<T, bool> querry, List<T> list)

{

return list.Where(querry).ToList();

}



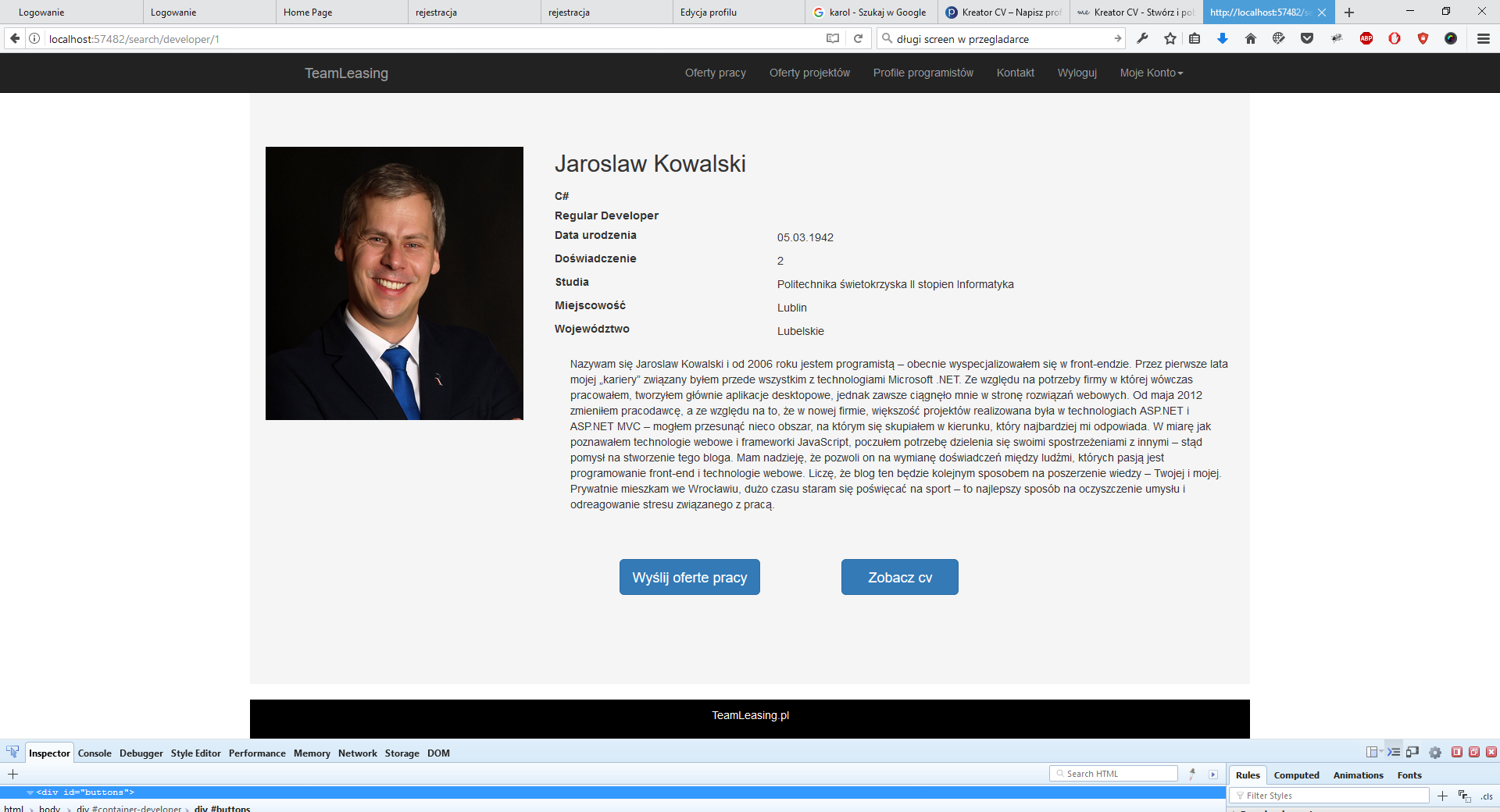
Rysunek 7.5 Wyszukiwarka ofert pracy

### Widok szczegółowy

Na rysunkach - *Rysunek 7.6* oraz *Rysunek 7.7* - przedstawiono widoki: profilu programisty oraz oferty pracy. W przypadku oferty pracy, użytkownik ma możliwość aplikowania na to stanowisko, a w przypadku profilu developera wysłanie mu propozycji pracy oraz przeglądanie jego CV, szczegóły tych funkcjonalności zostaną opisane w dalszych rozdziałach.



Rysunek 7.6 Widok szczegółowy oferty pracy



Rysunek 7.7 Widok szczegółowy profilu programisty

## Dodawanie nowych ofert pracy/projektów

Dodawanie nowych ofert pracy i projektów jest dostępne w panelu użytkownika po zalogowaniu się pracodawcy do serwisu. W oryginalnej wersji serwisu pracodawca nie ma żadnych ograniczeń liczby tworzonych ofert, w przyszłości można wprowadzić limity na tworzenie ogłoszeń, można również wprowadzić pakiety po wykupieniu których użytkownik dostanie możliwość wystawiania większej liczby ofert. Formularze w przypadku oferty pracy (*Rysunek 7.10*) i projektu (*Rysunek 7.9*), różnią się tylko niektórymi polami. Do formatowania tekstu ogłoszenia skorzystano z gotowego pluginu TinyMCE [23]. Jest to gotowy, niezależny edytor tekstu WYSIWYG (ang. what you see is what you get), napisany w języku JavaScript. Ten plugin, do standardowego pola textarea dodaje edytor, którego właściwości określone są w skrypcie inicjalizującym TinyMCE (Listing 7.7).

Listing 7.7 Inicjalizacja edytora TinyMCE

tinymce.init({

selector: "textarea",

height: 500,

plugins: [

"advlist autolink autosave link image lists charmap print preview hr anchor pagebreak",

"searchreplace wordcount visualblocks visualchars code fullscreen insertdatetime media nonbreaking",

"table contextmenu directionality emoticons template textcolor paste fullpage textcolor colorpicker textpattern"

],

toolbar1: "newdocument fullpage | bold italic underline strikethrough | alignleft aligncenter alignright alignjustify | styleselect formatselect fontselect fontsizeselect",

toolbar2: "cut copy paste | searchreplace | bullist numlist | outdent indent blockquote | undo redo | link unlink anchor image media code | insertdatetime preview | forecolor backcolor",

toolbar3: "table | hr removeformat | subscript superscript | charmap emoticons | print fullscreen | ltr rtl | visualchars visualblocks nonbreaking template pagebreak restoredraft",

Dodawanie oferty następuje dzięki metodzie CreateJob (Dodawanie oferty). Metoda ta znajduje się w UserManager. Standardowy UserManager, wbudowany w bibliotekę Identity daje możliwość rozszerzenia funkcjonalności, można wykorzystać dziedziczenie i w prosty sposób stworzyć własny Manager rozbudowany o swoje autorskie metody. Wiele metod do obsługi i zarządzania projektami, ofertami czy negocjacjami została właśnie zaimplementowana w takim rozszerzonym UserManager’rze.

Listing 7.8 Dodawanie oferty

public async Task<bool> CreateJob(CreateJobViewModel model, User user)

{

var job = \_mapper.Map<Job>(model);

job.IsHidden = false;

job.StatusForEmployee = Enums.JobStatusForEmployee.InProgress;

job.EmployeeUser = user.EmployeeUser;

var technology = await GetTechnology(w => w.Name == model.ChoosenTechnology);

job.Technology = technology.FirstOrDefault();

await \_teamLeasingContext.Jobs.AddAsync(job);

var result = await \_teamLeasingContext.SaveChangesAsync();

return Convert.ToBoolean(result);

}

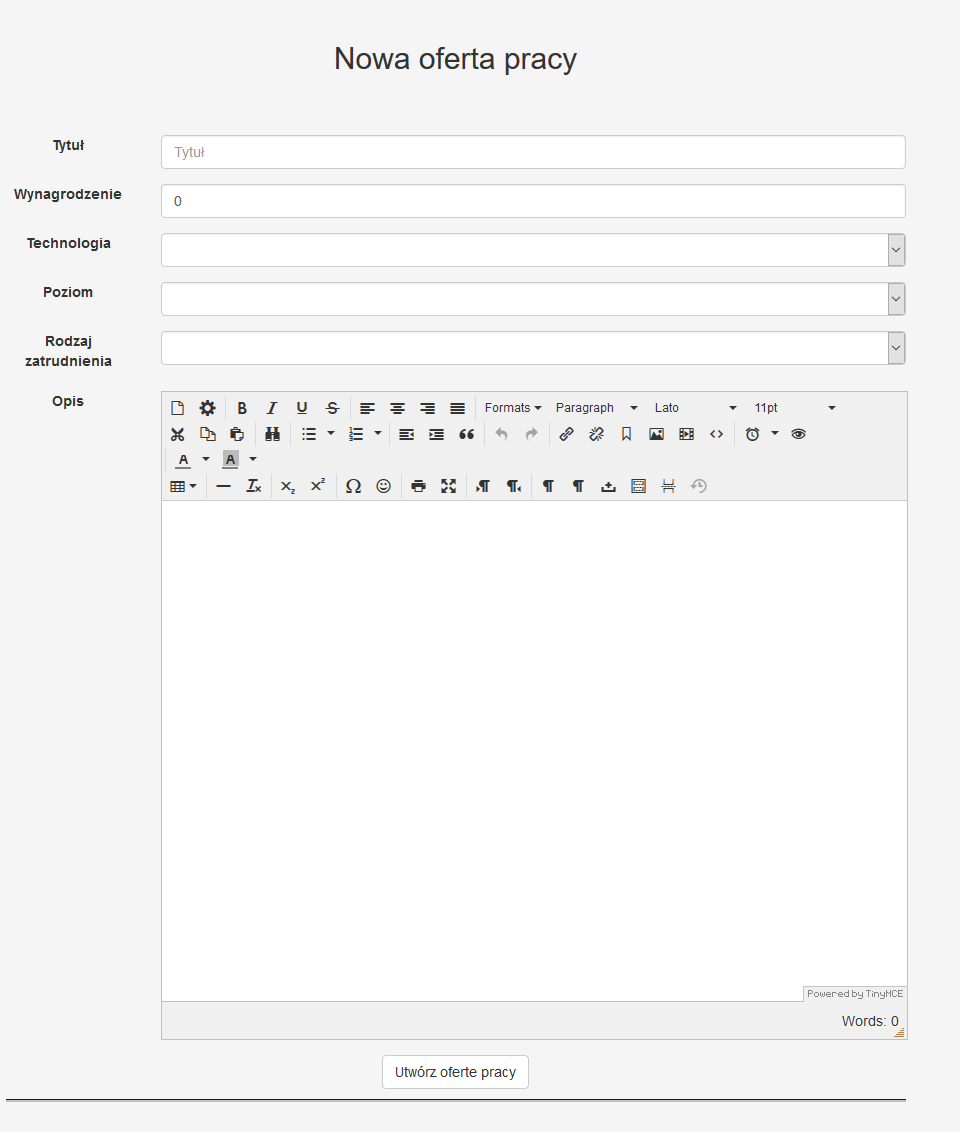
W momencie utworzenia oferty ustawiany jest status na InProgress. W panelu pracodawcy, w zakładce oferty pracy (*Rysunek 7.10*), od razu pojawia się kolejny wpis dotyczący nowo dodanego ogłoszenia.



Rysunek 7.8 Oferty pracy widoczne w panelu użytkownika



Rysunek 7.9 Widok tworzenia projektu



Rysunek 7.10 Widok tworzenia oferty pracy

## Składanie aplikacji przez programistę

Programista, po wybraniu interesującej go oferty pracy, może złożyć na to stanowiska aplikacje. Wysyłając aplikacje, metoda ApplyForJob (Listing 7.9) tworzy nowy wpis w tabeli DeveloperUserJob dla id obecnego użytkownika oraz id tej oferty pracy. Dodatkowo, zmieniany jest status tej aplikacji, na „Applying”, który będzie widoczny tylko dla Developera, w jego panelu użytkownika (). Właściciel tego ogłoszenia również zostanie poinformowany o fakcie aplikacji przez programistę, będzie mógł zarządzać tą aplikacją w jego profilu użytkownika ().

Listing 7.9 Wysyłanie aplikacji

public async Task<int> ApplyForJob(string userId, int jobId)

{

var developer = await GetDeveloperUser(w => w.UserId == userId)

.ContinueWith(t => t.Result.First());

if (!developer.Jobs.Any(a => a.JobId == jobId &&

a.StatusForDeveloper == Enums.JobStatusForDeveloper.Applying))

if (!developer.Jobs.Any(a => a.JobId == jobId))

{

var developerUserJob = new DeveloperUserJob

{

DeveloperUserId = developer.Id,

JobId = jobId,

StatusForDeveloper = Enums.JobStatusForDeveloper.Applying

};

await \_teamLeasingContext.DeveloperUserJob.AddAsync(developerUserJob);

return await \_teamLeasingContext.SaveChangesAsync();

}

else

{

developer.Jobs.First(f => f.JobId == jobId).StatusForDeveloper =

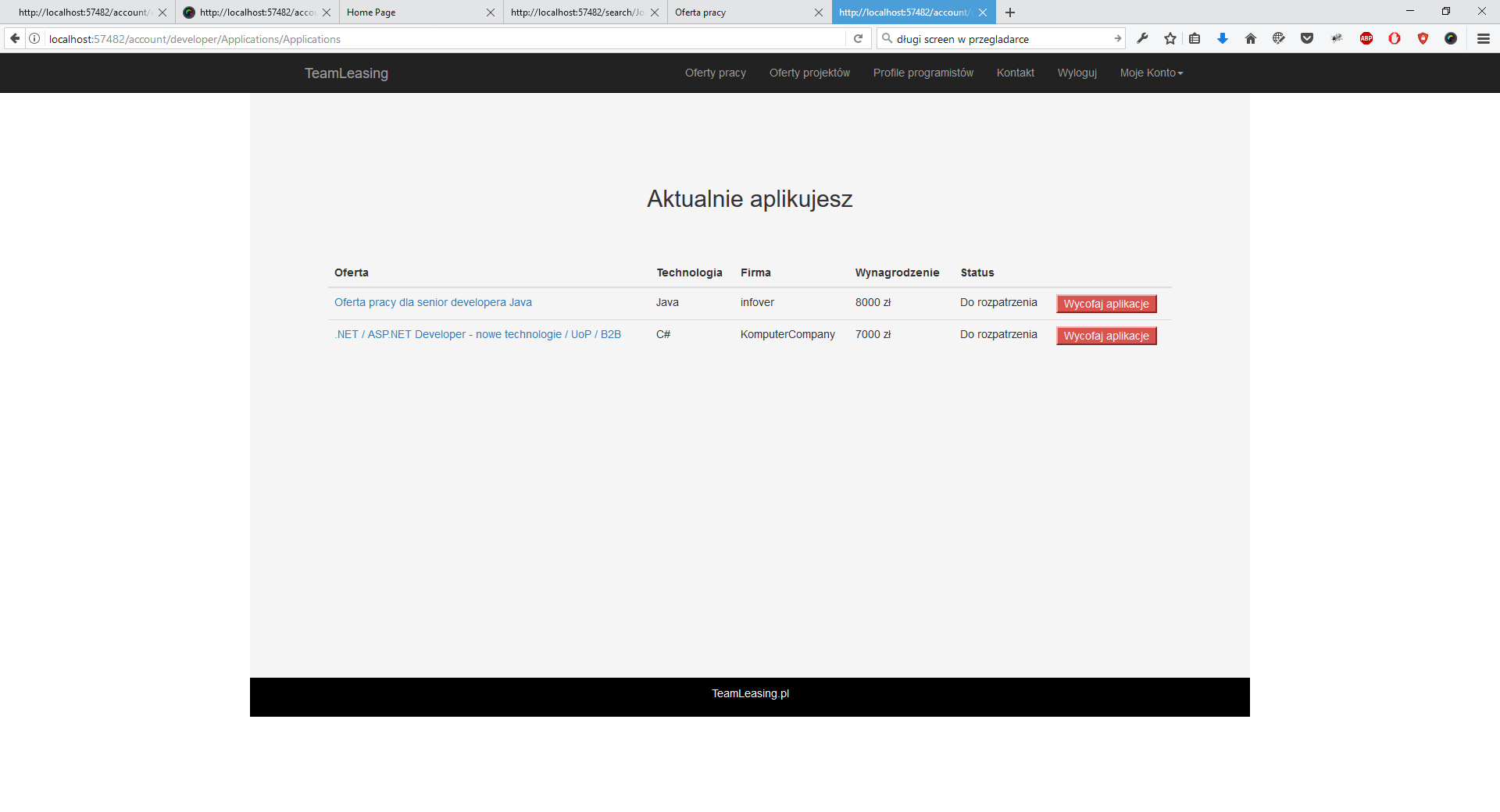
Enums.JobStatusForDeveloper.Applying;

return await \_teamLeasingContext.SaveChangesAsync();

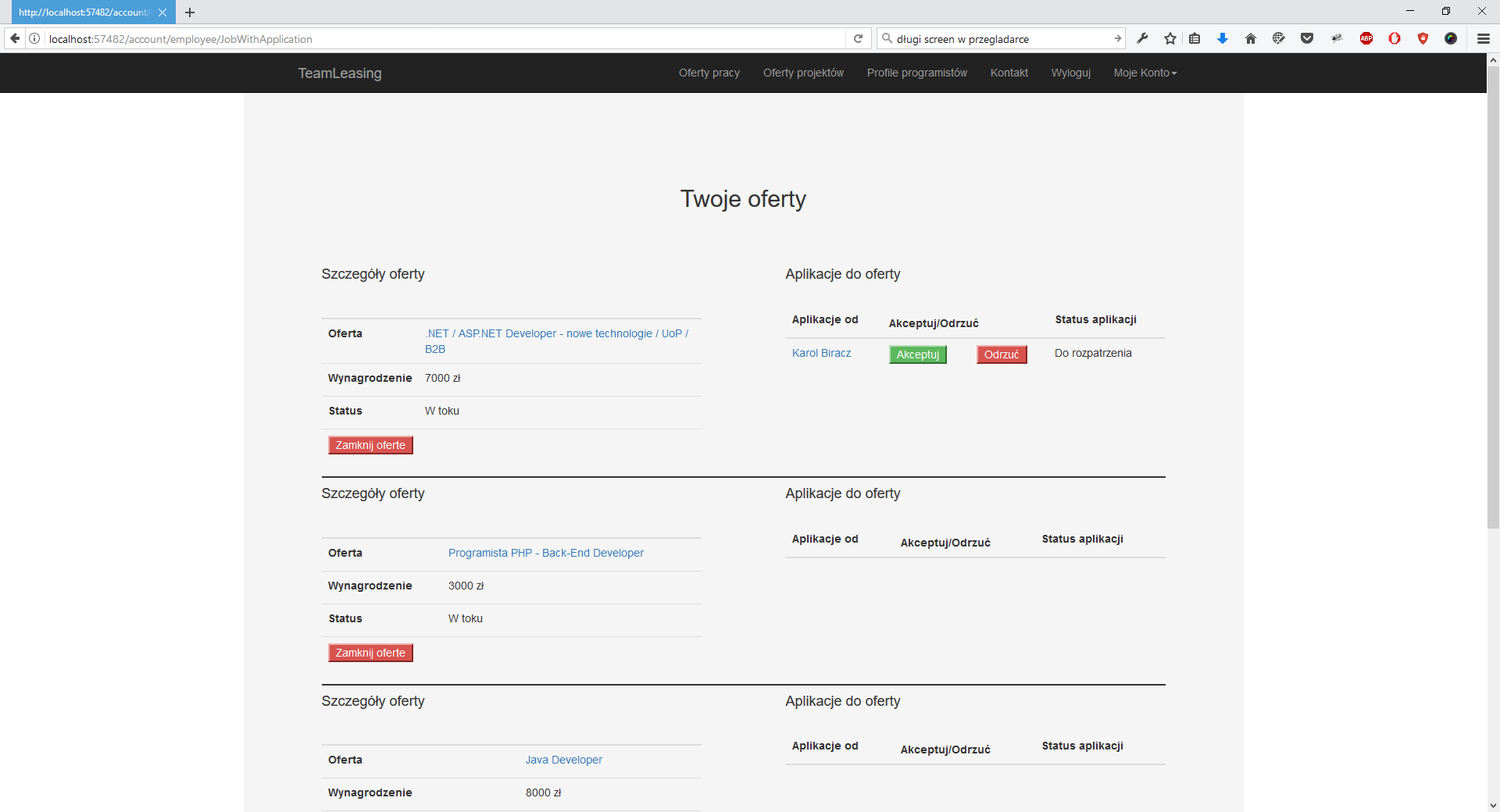
}

return -1;

}



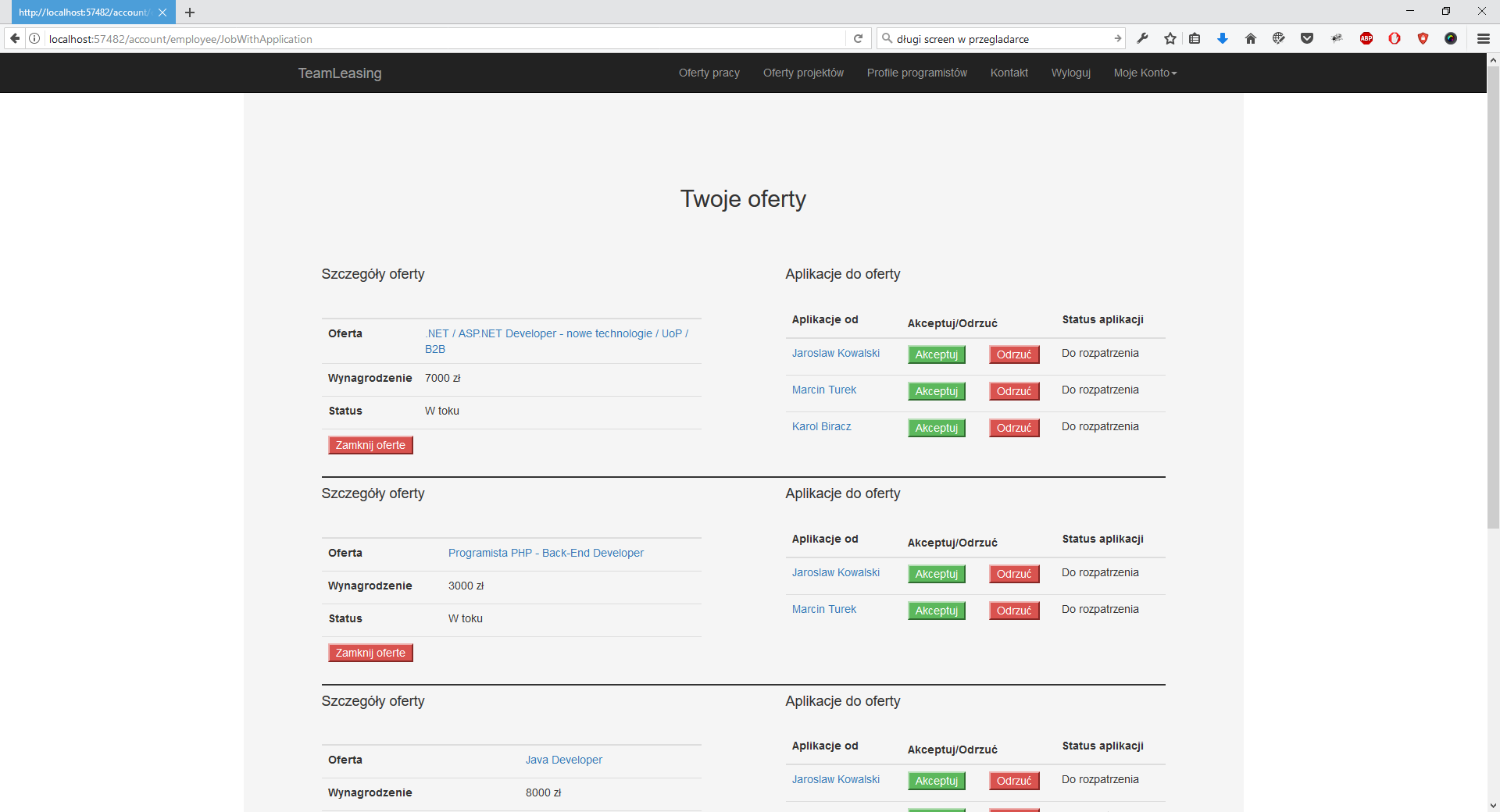
Rysunek 7.11 Panel programisty z jego aplikacjami



Rysunek 7.12 Panel pracodawcy z jego ofertami

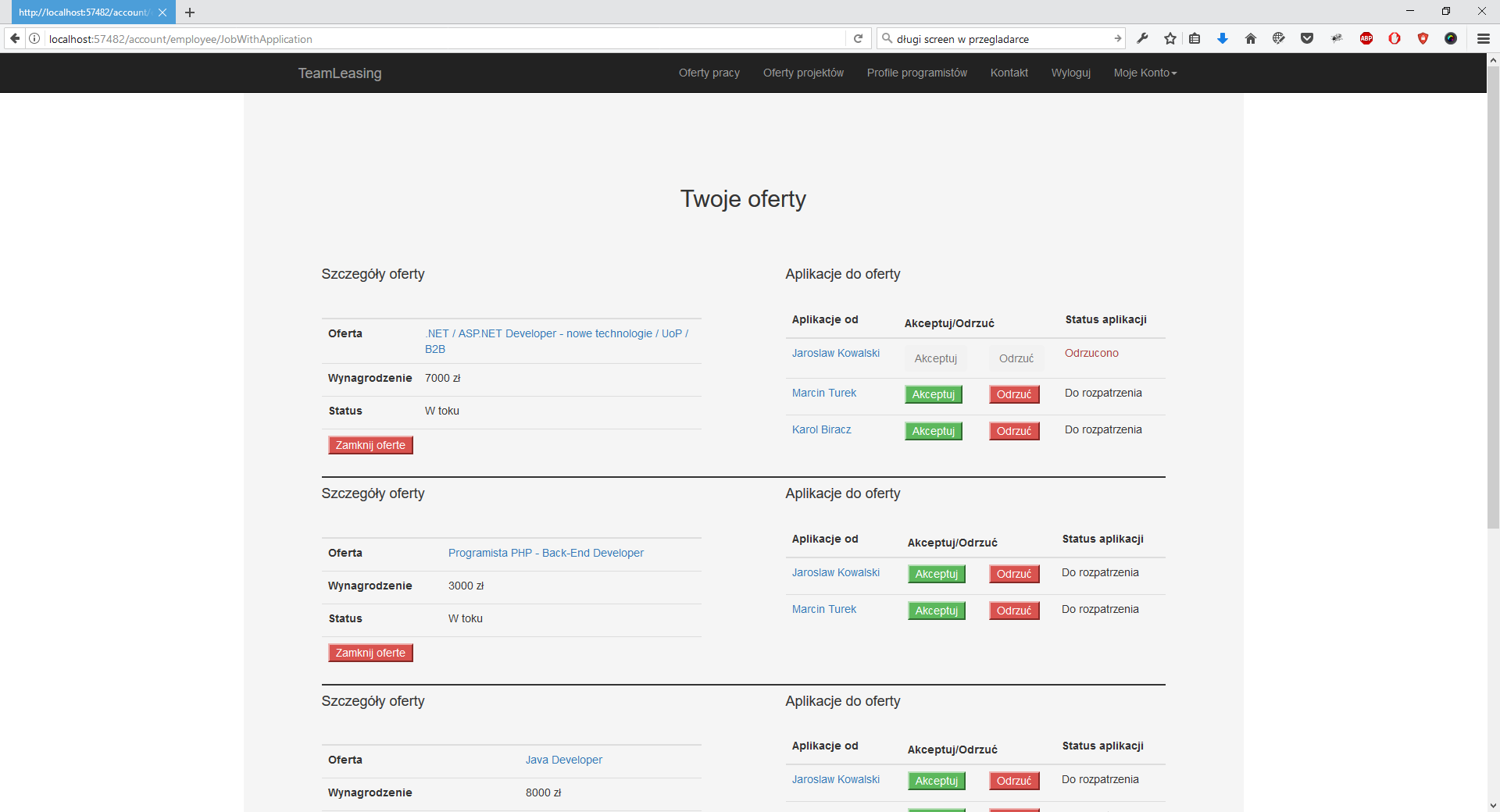
## Zarządzanie ofertami z panelu pracodawcy

Użytkownik, który ma role w serwisie „Pracodawca”, może zarządzać swoimi ofertami i aplikacjami do tej oferty, z poziomu swojego panelu użytkownika ()

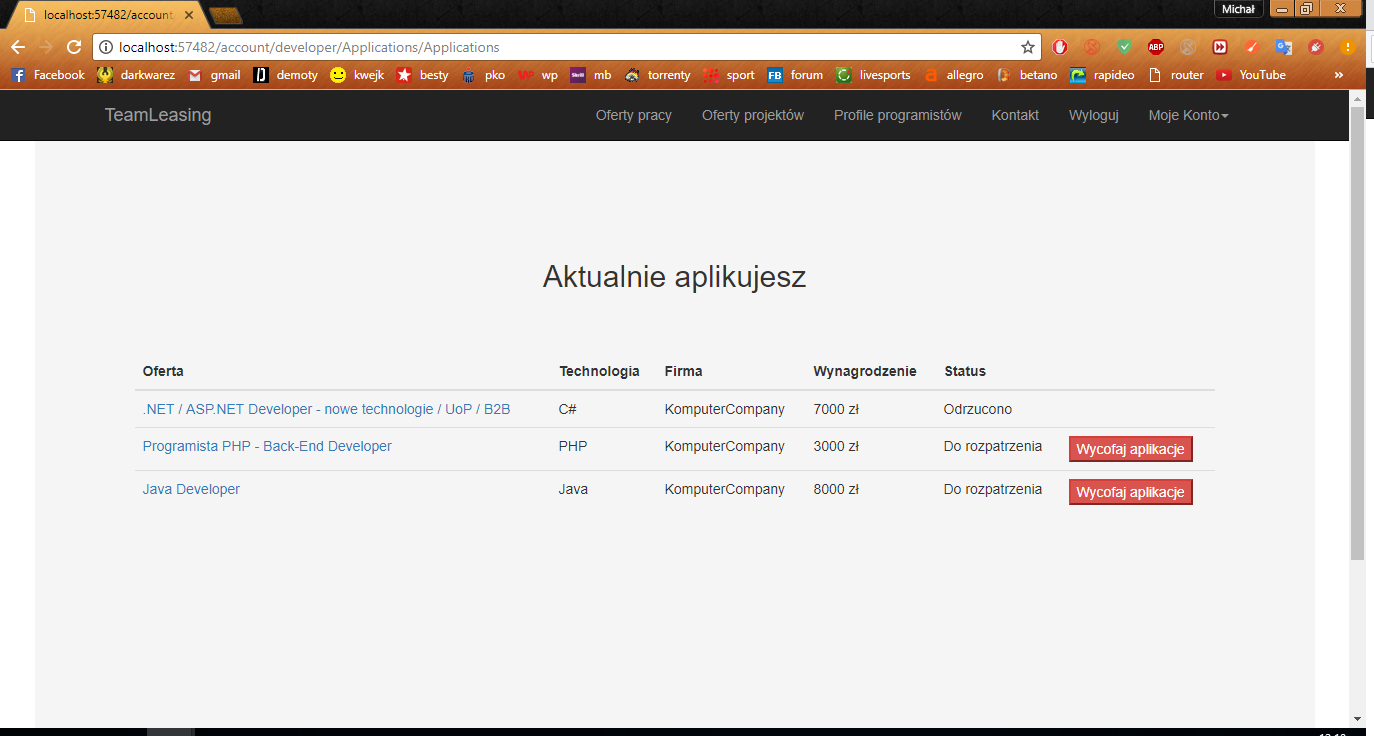


Rysunek 7.13 Zarządzanie ofertami i aplikacjami do tej oferty

Ma możliwość odrzucenia aplikacji, akceptacji lub po prostu może zamknąć swoje ogłoszenie. Po wyborze każdej z tych akcji, status oferty, czy też aplikacji ulegnie zmianie. W przypadku odrzucenia aplikacji, statusy aplikacji w panelu pracodawcy () jak i w panelu programisty (), ulegną zmianie.



Rysunek 7.14 Odrzucenie aplikacji - panel pracodawcy

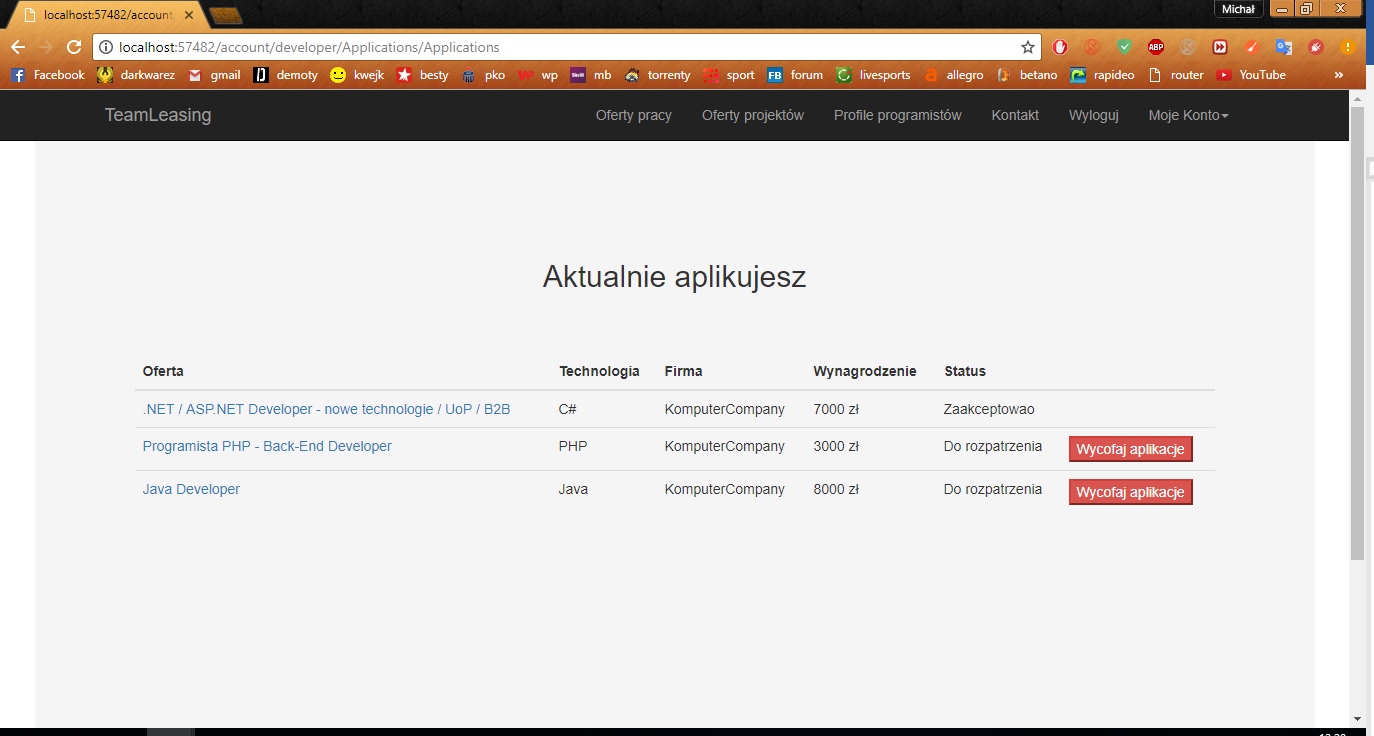


Rysunek 7.15 Odrzucenie aplikacji panel programisty

W przypadku akceptowania aplikacji, statusy również zmienią swoje wartości. Cała oferta zostanie automatycznie zamknięta oraz automatycznie zostanie wybrana zwycięska aplikacja (). Programista w swoim panelu również zostanie poinformowany o fakcie zaakceptowania jego aplikacji ().

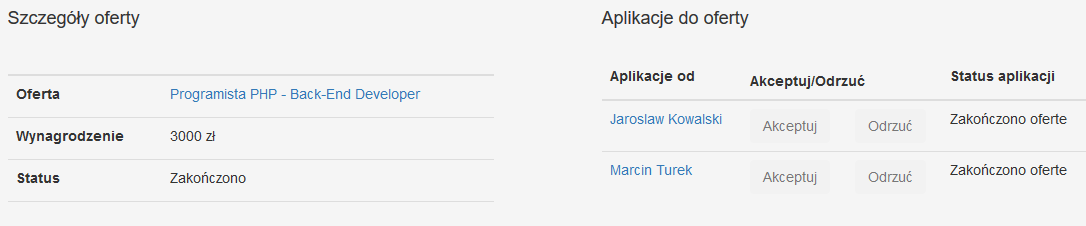


Rysunek 7.16 Akceptowanie aplikacji – panel pracodawcy

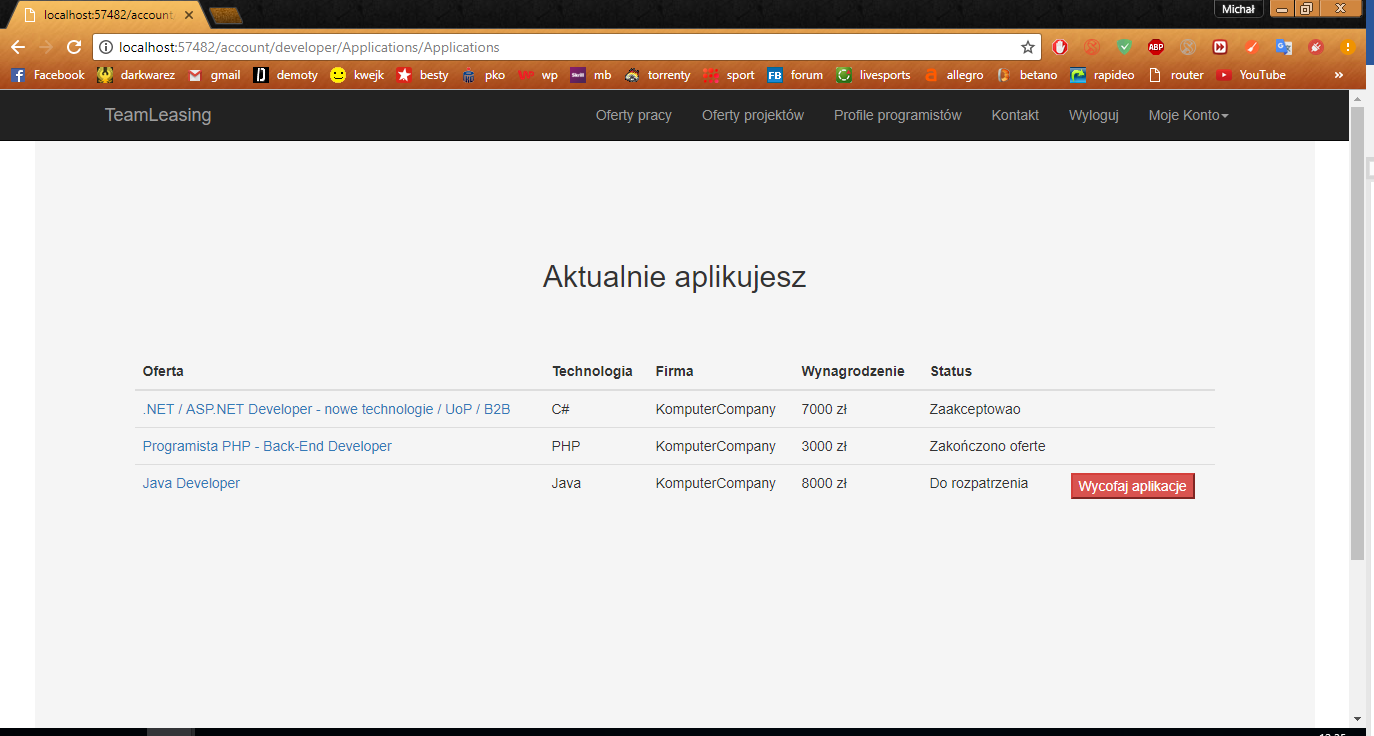


Rysunek 7.17 Akceptowanie aplikacji – panel programisty

W każdej chwili, pracodawca ma możliwość całkowitego zamknięcia swojego ogłoszenia, bez wybierania jakiejkolwiek aplikacji (). W panelu programisty automatycznie status zmieni się na „Zakończono ofertę” ().



Rysunek 7.18 Zamknięcie ogłoszenia – panel pracodawcy



Rysunek 7.19 Zamknięcie ogłoszenia – panel programisty

Na Listing 7.10, Listing 7.11, Listing 7.12 przedstawiono kod źródłowy metod, które zarządzają ofertami oraz aplikacjami. Wszystko, tak naprawdę sprowadza się do zamiany statusów oraz uaktualnieniu wpisów w bazie danych. W przypadku metody FinishJob oraz AcceptJobApplication istotny jest parametr „IsHidden”, ustawienie tego parametru na „true” wyklucza daną ofertę z rezultatów wyszukiwań w wyszukiwarce na stronie głównej. Zapobiegnie to niepożądanym sytuacjom, w którym jakaś ofertę będzie zamknięta, a ktoś będzie próbował na nią aplikować. Dostęp do niej będzie miał jedynie właściciel oferty.

Listing 7.10 Metoda odrzucająca aplikacje

public async Task<int> RejectJobApplication(int jobId, int developerId)

{

var developeUserJob = await GetDeveloperUsersJob(s => s.DeveloperUserId == developerId

&& s.JobId == jobId)

.ContinueWith(t => t.Result.FirstOrDefault());

developeUserJob.StatusForDeveloper = Enums.JobStatusForDeveloper.Rejected;

var result = await \_teamLeasingContext.SaveChangesAsync();

return result;

}

Listing 7.11 Metoda akceptująca aplikacje

public async Task<int> AcceptJobApplication(int jobId, int developerId)

{

var developeUserJob = await GetDeveloperUsersJob(s => s.DeveloperUserId == developerId

&& s.JobId == jobId)

.ContinueWith(t => t.Result.FirstOrDefault());

developeUserJob.StatusForDeveloper = Enums.JobStatusForDeveloper.Accepted;

developeUserJob.Job.IsHidden = true;

developeUserJob.Job.StatusForEmployee = Enums.JobStatusForEmployee.Approve;

var result = await \_teamLeasingContext.SaveChangesAsync();

return result;

}

Listing 7.12 Metoda zamykająca oferte

public async Task<int> FinishJob(int id)

{

var jobDao = await GetJob(w => w.Id == id);

var job = jobDao.FirstOrDefault();

job.IsHidden = true;

job.StatusForEmployee = Enums.JobStatusForEmployee.Finished;

foreach (var item in job.DeveloperUsers)

item.StatusForDeveloper = Enums.JobStatusForDeveloper.Finished;

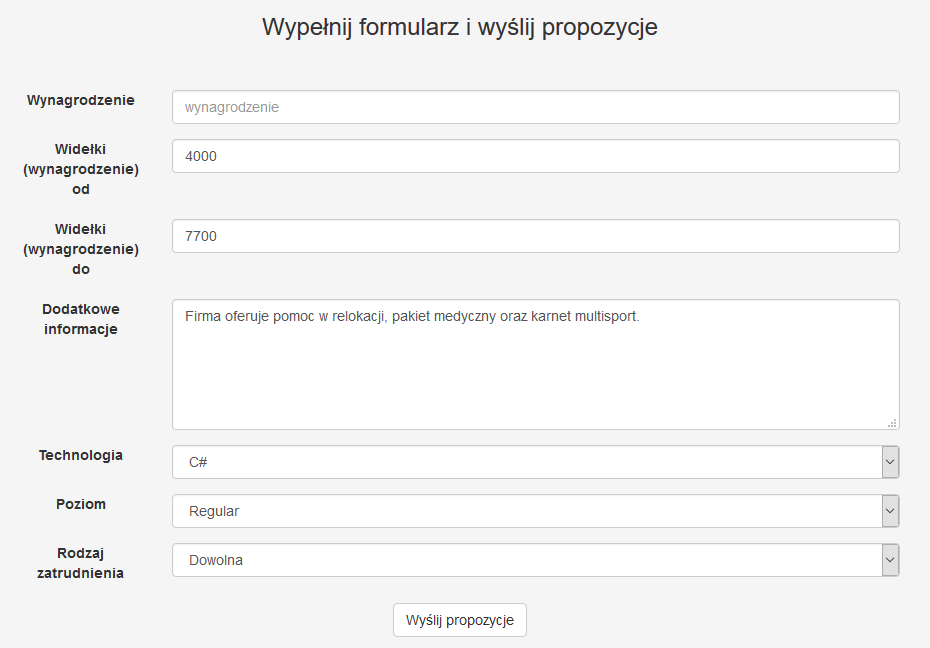
\_teamLeasingContext.Update(job);

return await \_teamLeasingContext.SaveChangesAsync();

}

## Propozycje pracy

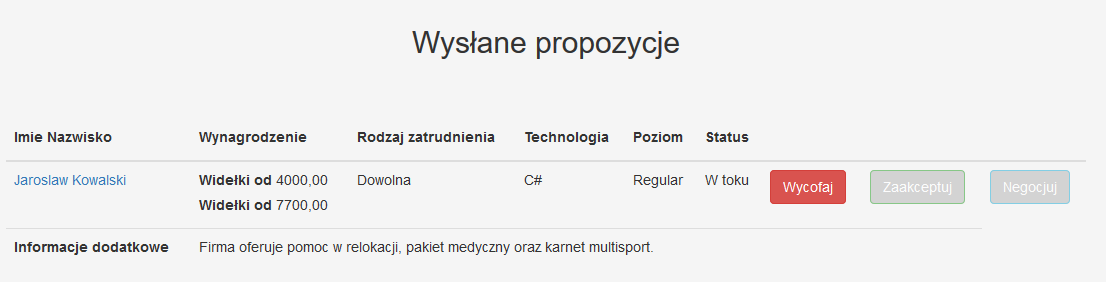
Każdy pracodawca zarejestrowany w serwisie, aby znaleźć pożądanego programistę, ma dwie możliwości: może biernie czekać, aż kandydaci sami złożą aplikacje do jego ogłoszenia lub może na własną rękę szukać pracownika i wysyłać mu propozycje pracy, wraz z określonymi warunkami zatrudnienia. Na późniejszym etapie, możliwe są również, negocjacje pierwotnej propozycji złożonej programiście przez pracodawcę. Po wybraniu profilu programisty, użytkownik ma możliwość zobaczyć cv kandydata lub otworzyć formularz z propozycją pracy (Rysunek 7.20).



Rysunek 7.20 Formularz z propozycją pracy

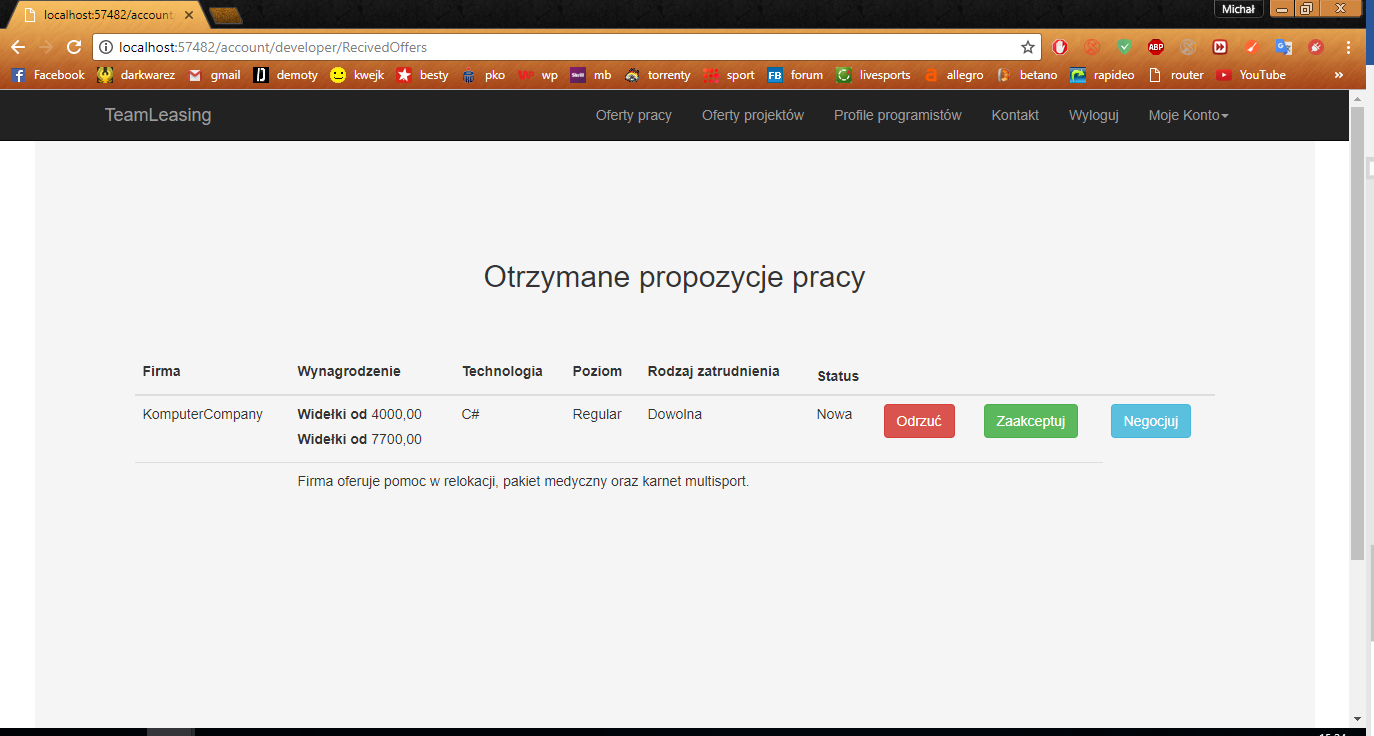
Pracodawca podaje podstawowe warunki zatrudnienia, takie jak wynagrodzenie (w formie stałego wynagrodzenia lub też może podać widełki płacowe ), rodzaj zatrudnienia, technologie czy proponowany rodzaj stanowiska. Po zatwierdzeniu i wysłaniu propozycji

Automatycznie pojawi się kolejny wpis w panelu pracodawcy w zakładce „Wysłane propozycje” .Jak widać na *Rysunek 7.21*, nowa propozycja pracy ma status „w toku” i czeka na odpowiedź developera, dlatego też zablokowane są możliwości zaakceptowania i negocjowania warunków, na tym etapie pracodawca ma możliwość jedynie wycofania propozycji.



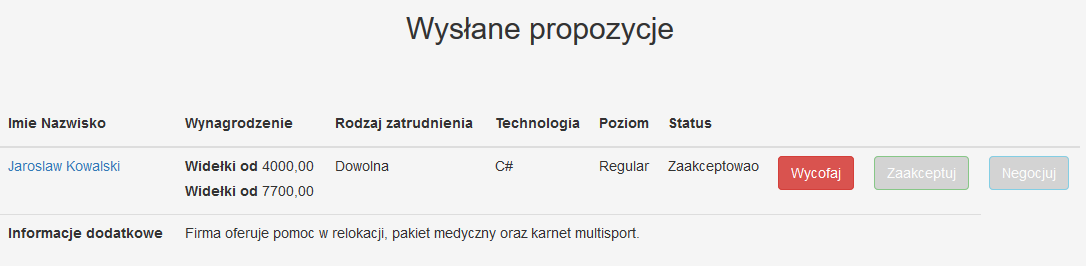
Rysunek 7.21 Wysłane propozycje pracy – panel pracodawcy

W panelu programisty pojawia się natomiats informacja o otrzymanej propozycji pracy (*Rysunek 7.22*). Ma on do wybory jedną z trzech możliwości: odrzucenie, zaakceptowanie lub negocjowanie oferty



Rysunek 7.22 Otrzymane propozycje pracy – panel programisty

W przypadku odrzucenia (Rysunek 7.24) lub akceptowania (*Rysunek 7.23*), rzecz ma się podobnie jak w poprzednich rozdziałąch opisywany proces składania aplikacji oraz ich późniejszego akceptowania bądź odrzucania. Implementacja tej funkcjonalności jest bliźniaczo podobna do funkcjonalości odrzucania i akceptowania aplikacji, rozwiązanie to zostało dobrze omówione w poprzednich podrozdziałach, więc nie ma sensu tego powtarzać.

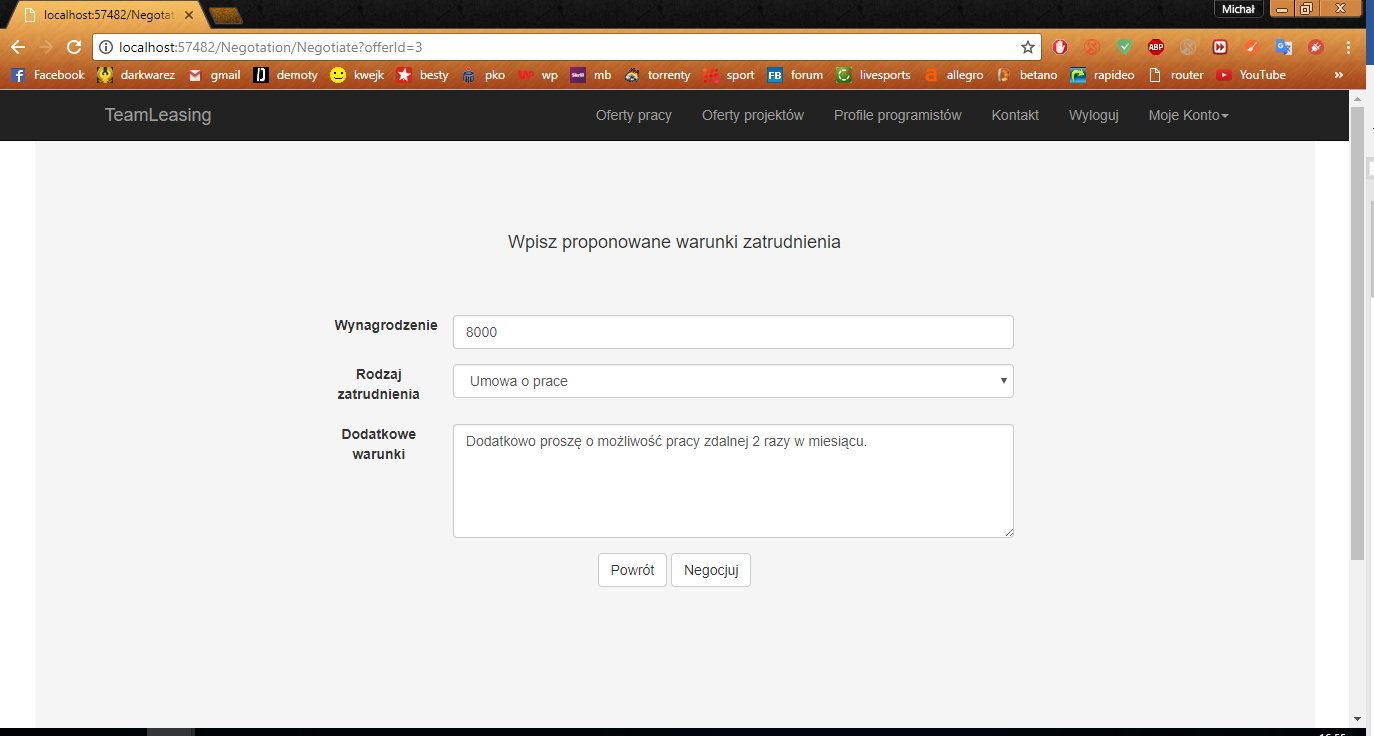


Rysunek 7.23 Wysłane propozycje – zaakceptowanie propozycji



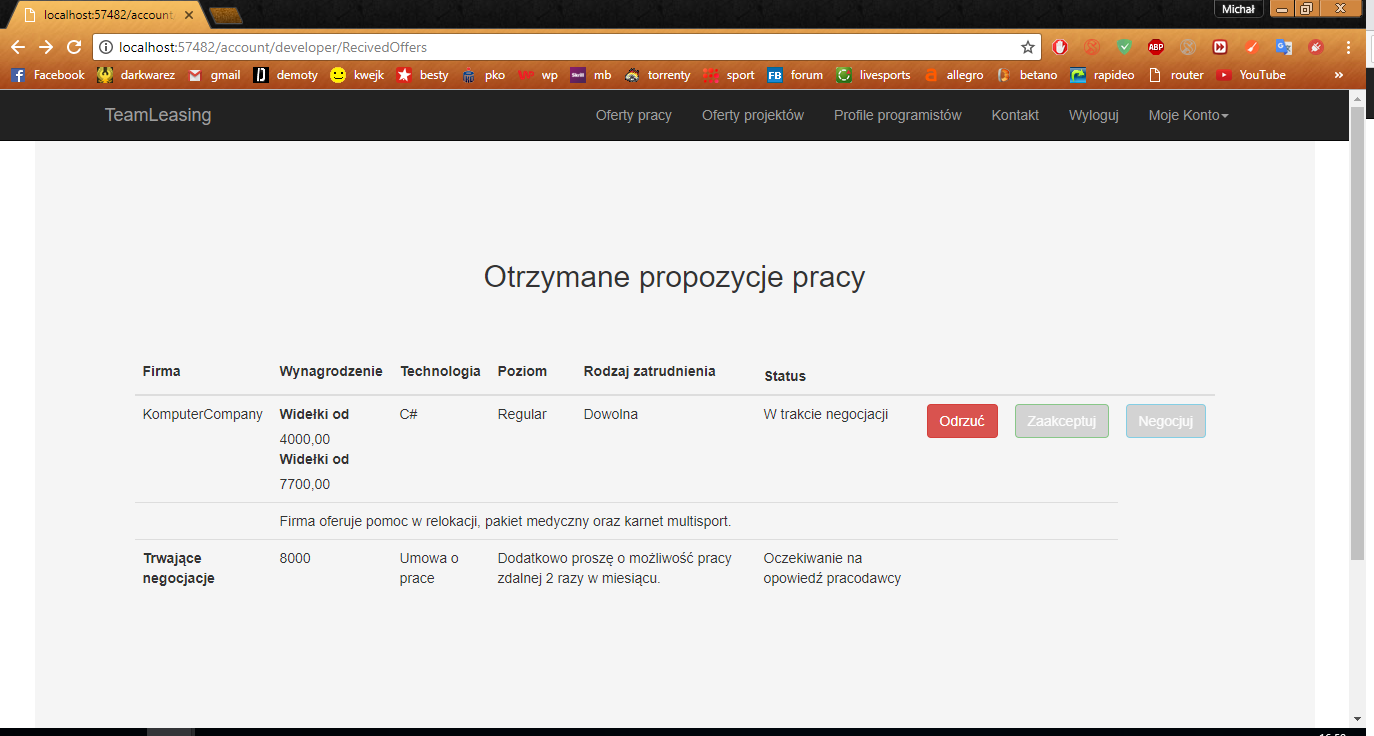
Rysunek 7.24 Wysłane propozycje – odrzucenie propozycji

Głębszego omówienia natomiast wymaga funkcjonalność negocjowania wysłanej propozycji pracy przez pracodawcę do developera. Po wybraniu przez programistę w zakładce „otrzymane oferty pracy” opcje „Negocjuj”, załaduje się formularz służący do negocjacji propozycji (Rysunek 7.25).



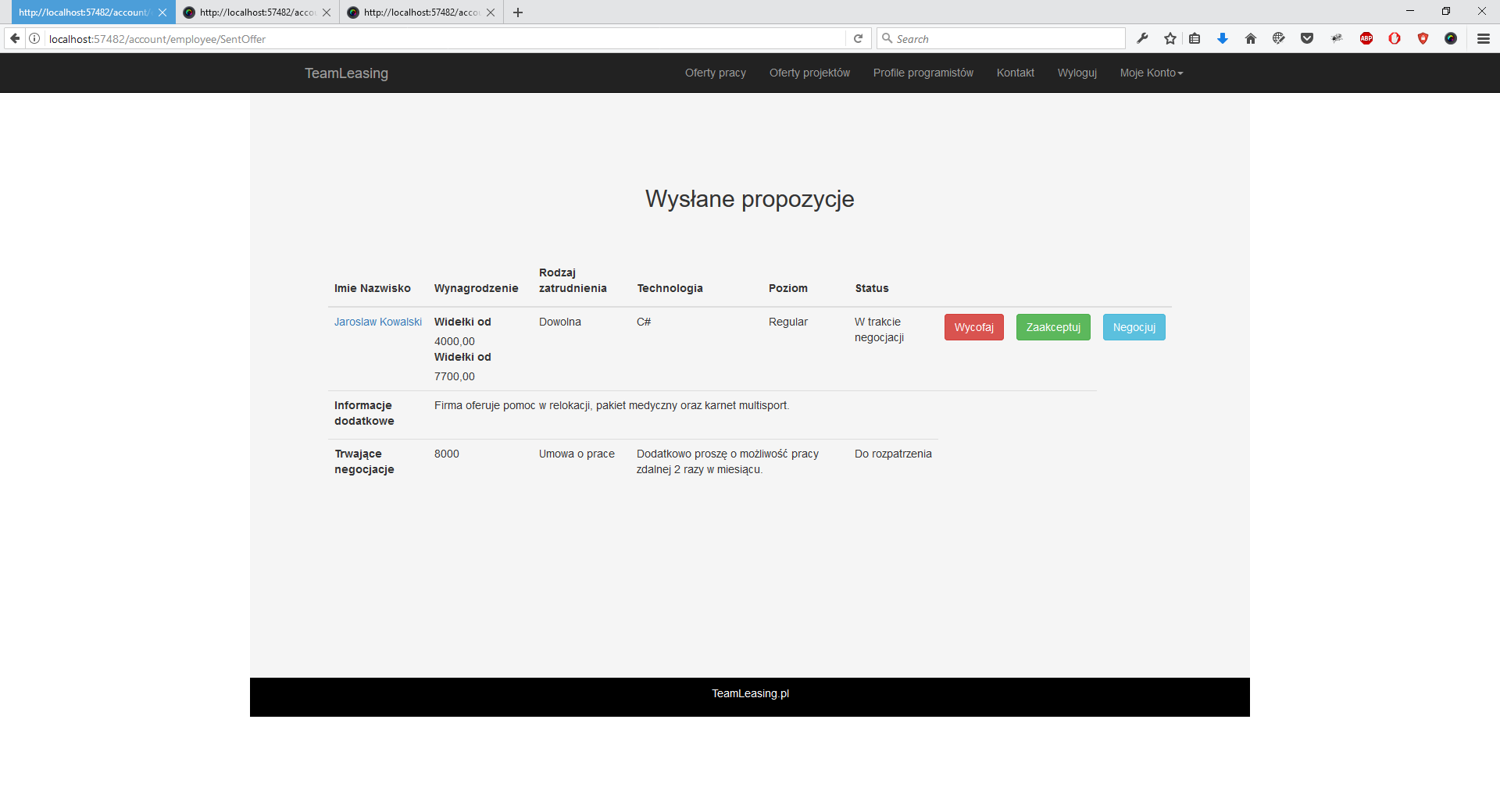
Rysunek 7.25 Formularz negocjacyjny

Po naciśnięciu przycisku „Negocjuj”, automatycznie zostanie w panelu dodana informacja o trwającej negocjacji i jej statusie (Rysunek 7.26).



Rysunek 7.26 Panel programisty – informacja o negocjacjach

W takiej sytuacji, pracodawca musi wykonać kolejny ruch, on również może renegocjować warunki zaproponowane przez potencjalnego pracownika (Rysunek 7.27).



Rysunek 7.27 Trwające negocjacje – panel pracodawcy

Metoda odpowiedzialna za dodawanie negocjacji do bazy danych, oraz aktualizowanie statusów tych negocjacji znajduje się na Listing 7.13

Listing 7.13 Metoda odpowiedzialna za realizacje negocjacji.

public async Task<int> AddOrUpdateNegotiation(Negotiation negotiation, int offerId,

Enums.NegotiationStatus developerStatus,

Enums.NegotiationStatus employeeStatus)

{

negotiation.StatusForDeveloper = developerStatus;

negotiation.StatusForEmployee = employeeStatus;

var offer = await FindOffer(offerId);

if (offer.Negotiation == null)

{

await \_teamLeasingContext.Negotiation.AddAsync(negotiation);

}

else

{

\_teamLeasingContext.Negotiation.Remove(offer.Negotiation);

offer.Negotiation = negotiation;

await \_teamLeasingContext.Negotiation.AddAsync(offer.Negotiation);

}

offer.StatusForEmployee = Enums.OfferStatus.Negotiation;

offer.StatusForDeveloper = Enums.OfferStatus.Negotiation;

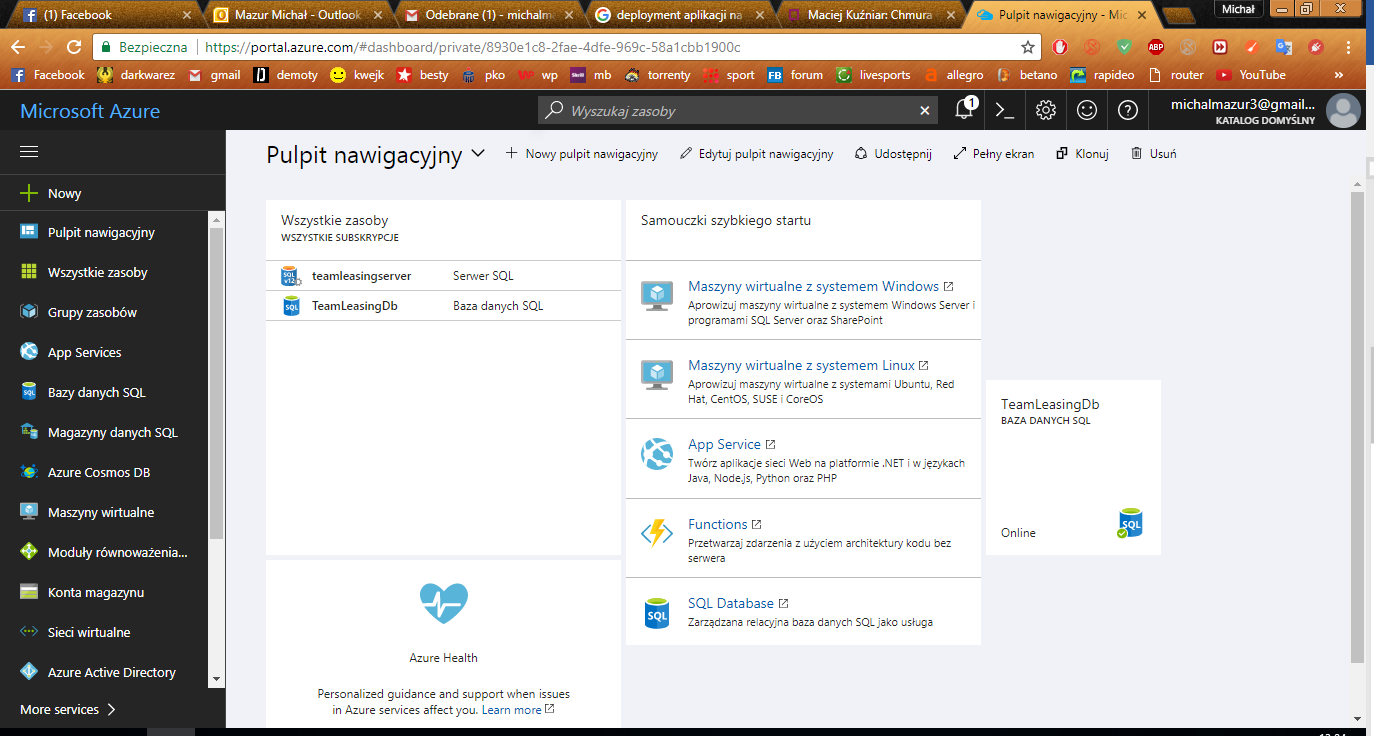
var returnResult = await \_teamLeasingContext.SaveChangesAsync();

return returnResult;

}

# Przenoszenie aplikacji do chmury obliczeniowej

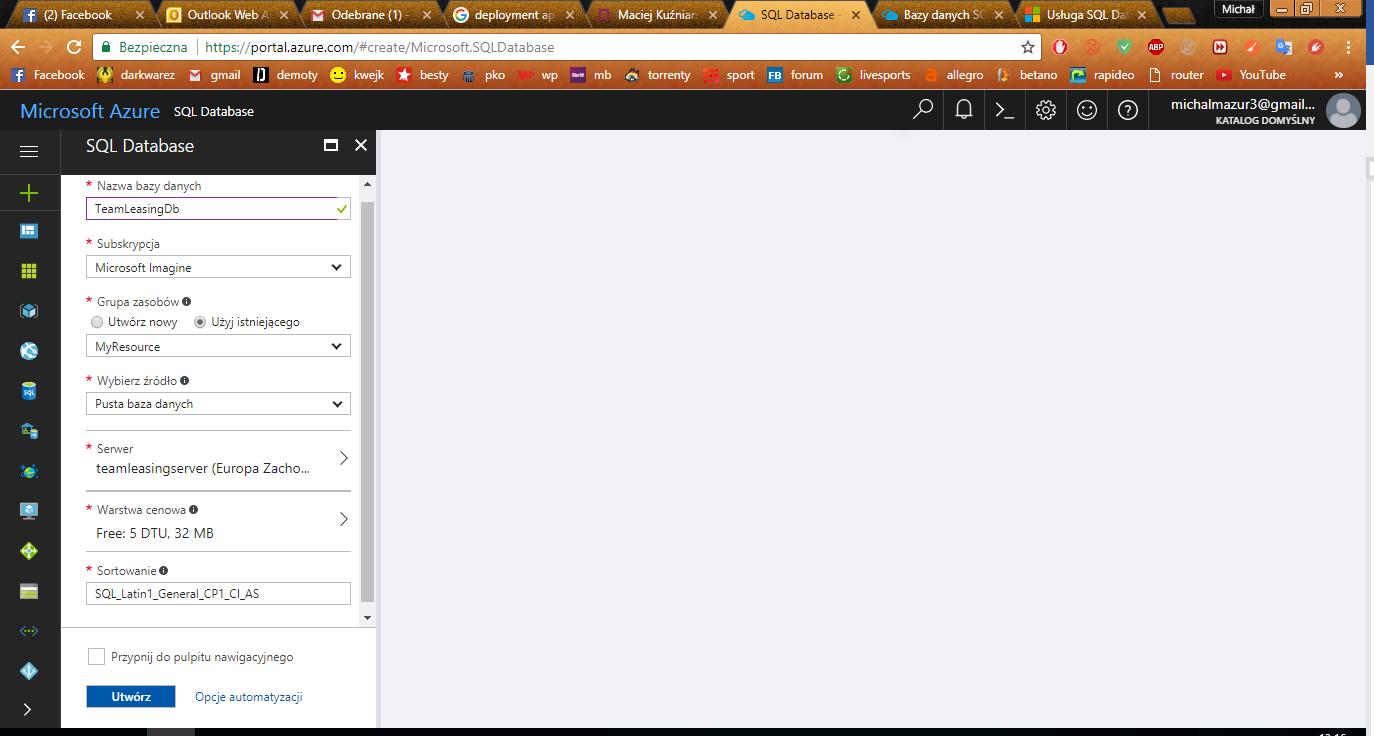
Proces developmentu aplikacji kończony jest zazwyczaj wystawieniem aplikacji do ogólnego dostępu. Można wybrać do tego celu prywatne wirtualne serwery lub chmurę. W przypadku pracy magisterskiej aplikacja w wersji testowej zostanie przeniesione na chmure Microsoft Azure. Dzięki subskrypcji na zasadzie licencji studenckiej DreamSpark, jest możliwość przeniesienia nie tylko samej aplikacji, ale też utworzenie różnego rodzaju innych zasobów, takich jak serwery czy bazy danych. Do obsługi swojego konta najlepiej skorzystać z portalu, który jest dostępny pod adresem portal.azure.com (*Rysunek 8.1*).



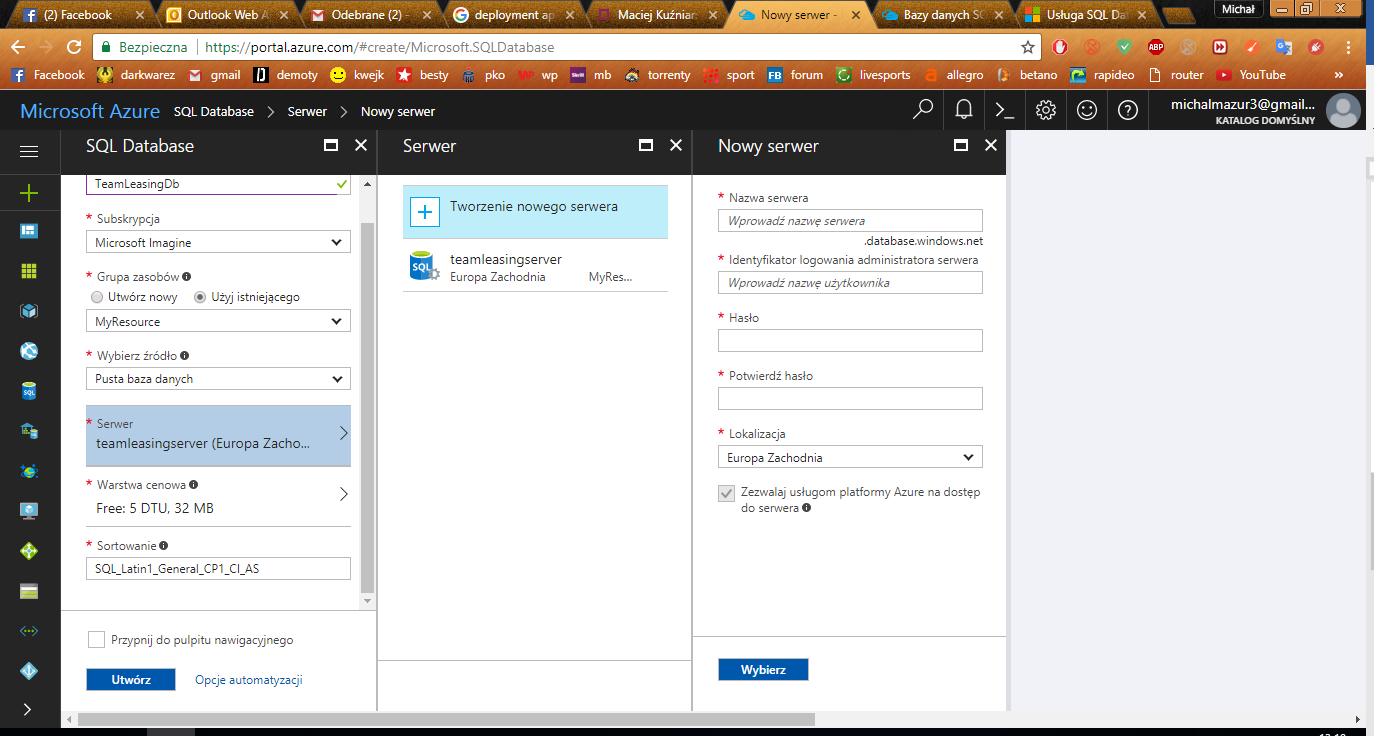
Rysunek 8.1 Ekran główny portalu Azure

## Baza danych SQL

Pierwszy krokiem jest utworzenie zasobu jakim jest baza danych. Tworzenie bazy danych w portalu nie jest skomplikowane, sprowadza się do wybrania z głównej zakładki „Bazy danych SQL” , a następnie do określenia podstawowych parametrów (*Rysunek 8.2*). Użytkownik określa nazwę bazy danych, subskrypcję – w tym przypadku można skorzystać tylko z subskrypcji Microsoft Imagine – zasób oraz serwer, który również trzeba utworzyć (Rysunek 8.3). Subskrypcja DreamSpark oferuje jedynie 5 jednostek DTU [24] (Database Transaction Units ) oraz 32 MB współdzielonego magazynu. Na potrzeby aplikacji magisterskiej, konto podstawowe w zupełności pokryje potrzeby serwisu.

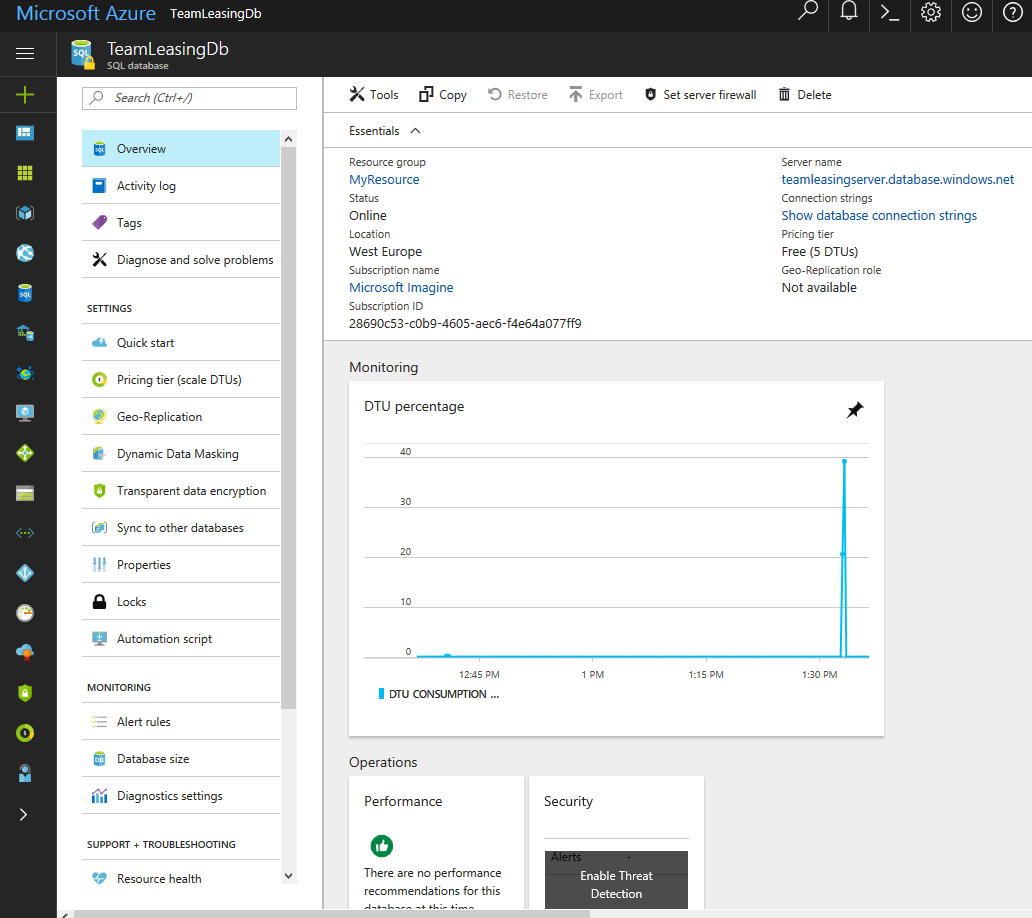


Rysunek 8.2 Parametry tworzonej bazy danych



Rysunek 8.3 Tworzenie serwera

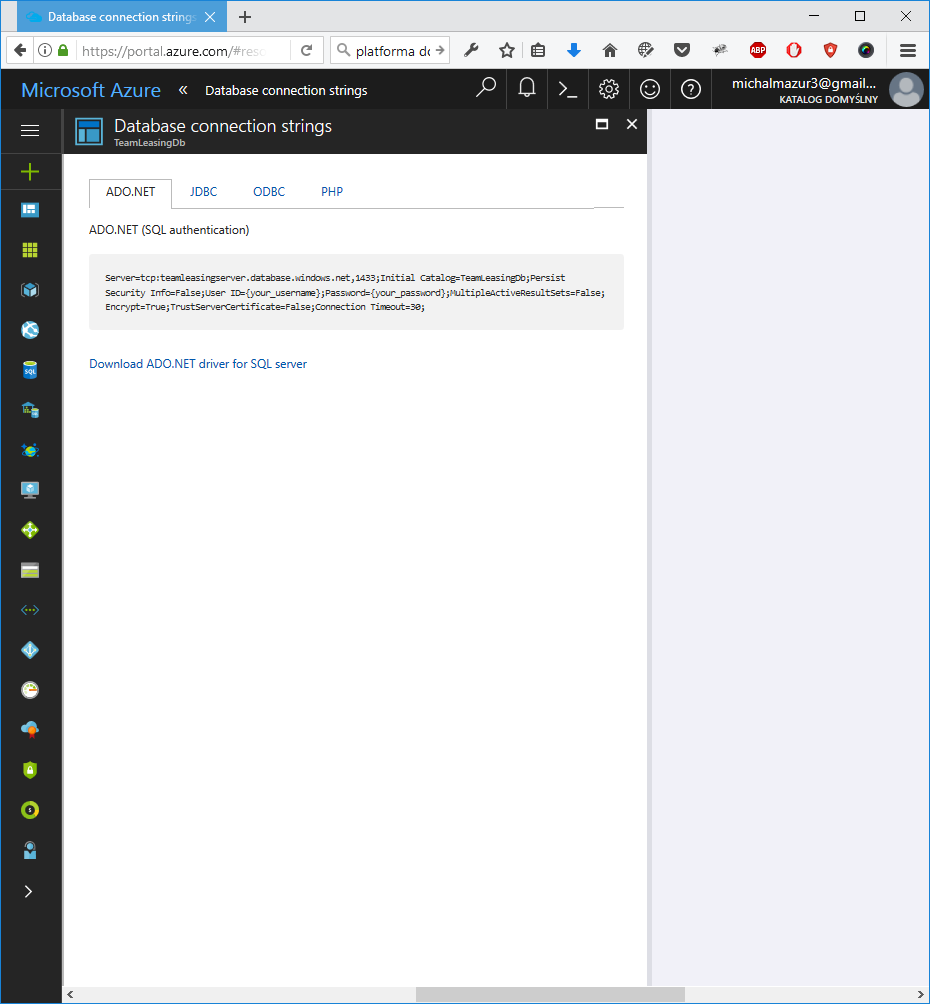
Po utworzeniu bazy danych, użytkownik ma dostęp do narzędzi zarządzających tym zasobem (*Rysunek 8.4*). Dostępne są różnego rodzaje narzędzia monitorujące, ustawienia firewall’u, czy integracji z aplikacjami. Z poziomu tego panelu, w łatwy sposób można też zwiększyć zasoby w przypadku potrzeby użycia większej mocy obliczeniowej.



Rysunek 8.4 Panel do obsługi bazy danych

### Połączenie z bazą danych w aplikacji

Kolejnym krokiem będzie zmiana magazynu danych w kodzie aplikacji. Do tej pory, czyli w wersji developerskiej, aplikacja korzystała z lokalnego serwera IIS czy z lokalnej bazy danych SQL. Pierwszą czynnością jaka należy teraz wykonać, to podmiana ConnectionString’a bazy danych. Connection String do lokalnej bazy danych należy zastąpić wartością dostępną w sekcji zarządzania bazą danych w portalu Azure (*Rysunek 8.5*).



Rysunek 8.5 Connection string do bazy danych w chmurze Azure

Całą wartość należy skopiować i dodać wpis w pliku konfiguracyjnym (*Listing 8.1*) oraz w kodzie aplikacji zmienić połączenie, tak aby kontekst bazy danych łączył się z bazą danych w chmurze Azure (*Listing 8.2*).

Listing 8.1 Plik konfiguracyjny z wartościami ConnectionStrings

{

"ConnectionStrings": {

"TeamLeasingConnectionStringLocal": "Data Source=DESKTOP-K9TOQB7;Initial Catalog=TeamLeasing;Integrated Security=True",

"TeamLeasingConnectionStringAzure": "Server=tcp:teamleasingserwer.database.windows.net,1427;Initial Catalog=TeamLeasingDb;Persist Security Info=False;User ID=Mazur;Password=Admin123$;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;"

}

}

Listing 8.2 Ustawienie ConnetionString w metodzie konfiguracyjnej kontekstu bazy danych

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

base.OnConfiguring(optionsBuilder);

optionsBuilder.UseSqlServer(\_config["ConnectionStrings:TeamLeasingConnectionStringAzure"]);

}

### Zainicjalizowanie bazy danych danymi testowymi

Po poprawnym utworzenia połączenia z magazynem danych, należy sprawdzić czy mamy dostęp do bazy SQl z aplikacji. Na Listing 8.3 znajduję się fragment kodu, który inizjalizuje baze danych, danymi testowymi.

Listing 8.3 Fragment kodu inicjalizującego bazę danych z przykładowymi danymi

private async Task DeveloperUser()

{

if (\_manager.FindByEmailAsync("jaroslaw@gmail.com").Result == null)

{

User user = new User()

{

Email = "jaroslaw@gmail.com",

UserName = "jarekKow",

PhoneNumber = "2342342523",

};

var result = await \_manager.CreateAsync(user, "Qazqaz12#");

if (result.Succeeded)

{

DeveloperUser dev = new DeveloperUser()

{

Name = "Jaroslaw ",

BirthDate = new DateTime(1942, 03, 05),

City = "Lublin",

Level = Enums.Level.Regular,

Experience = 2,

Province = Enums.Province.Lubelskie.ToString(),

Surname = "Kowalski",

University = "Politechnika świetokrzyska II stopien Informatyka",

IsFinishedUniversity = Enums.IsFinishedUniversity.Finished,

Technology = await \_context.Technologies.FindAsync(2),

Photo = "/uploadfile/photo/Kowalski\_Jaroslaw.jpg",

UserId = \_context.Users.Find(user.Id).Id,

Cv = "/uploadfile/cv/cv.pdf",

About = "Jestem z zawodu i z zamiłowania programistą. Czuję się pewnie pisząc w JavaScripcie i TypeScripcie (głównie Node.js, choć także po stronie przeglądarki), w Javie, a na wcześniejszych etapach kariery programowałem również w PHP. Chętnie uczę się nowych technik programowania, szybko przyzwyczajam się do nowości w światku technologii. Eksperymentuję z nowymi narzędziami i językami programowania. Od czasu do czasu prowadzę prelekcje na spotkaniach meet.js w Gdańsku, zdarzyło mi się też prowadzić kilka warsztatów dotyczących programowania sterowanego testami(TDD).Jak mam natchnienie i nadmiar czasu, to dzielę się doświadczeniami i przemyśleniami na niniejszym blogu lub na kanale YouTube."

};

await \_context.DeveloperUsers.AddAsync(dev);

await \_context.SaveChangesAsync();

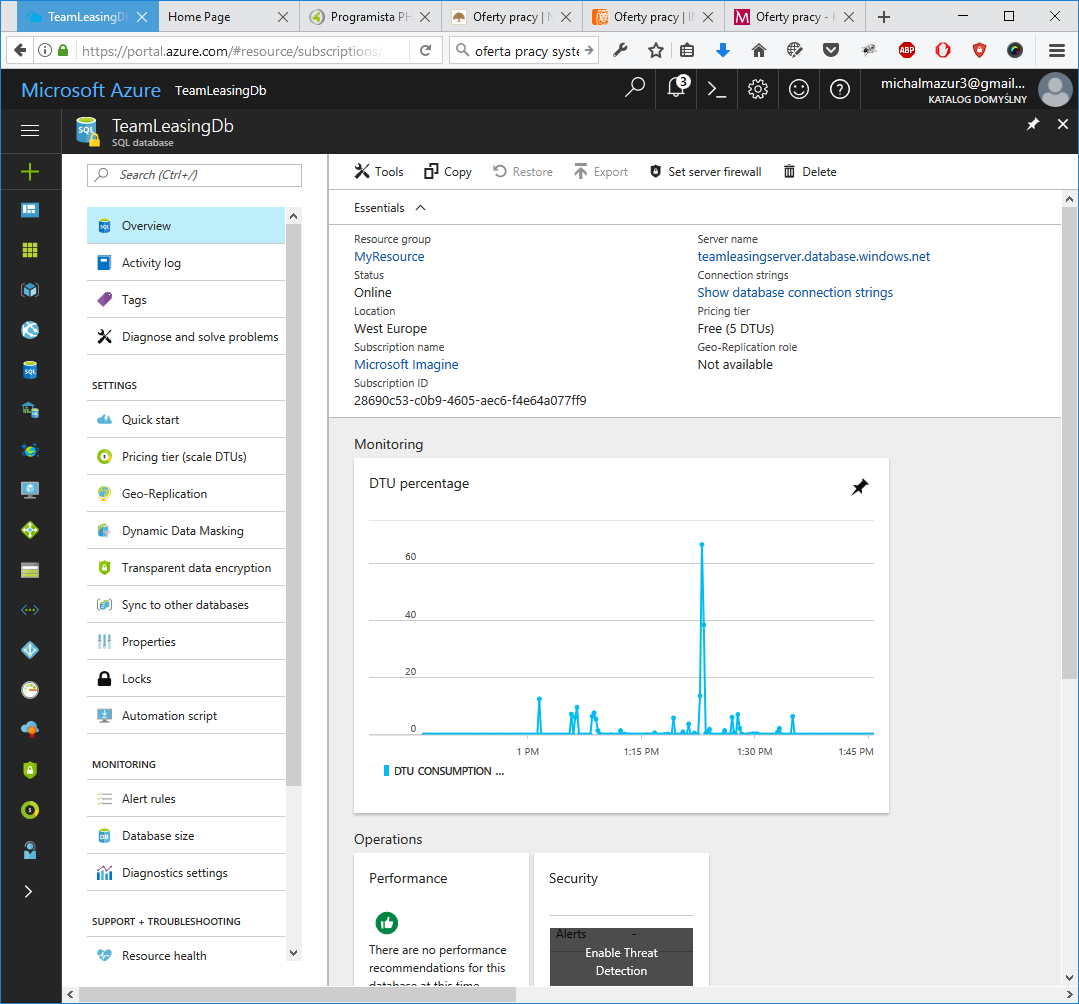
await \_manager.AddToRoleAsync(\_context.Users.Find(user.Id), Roles.Developer.ToString());

}

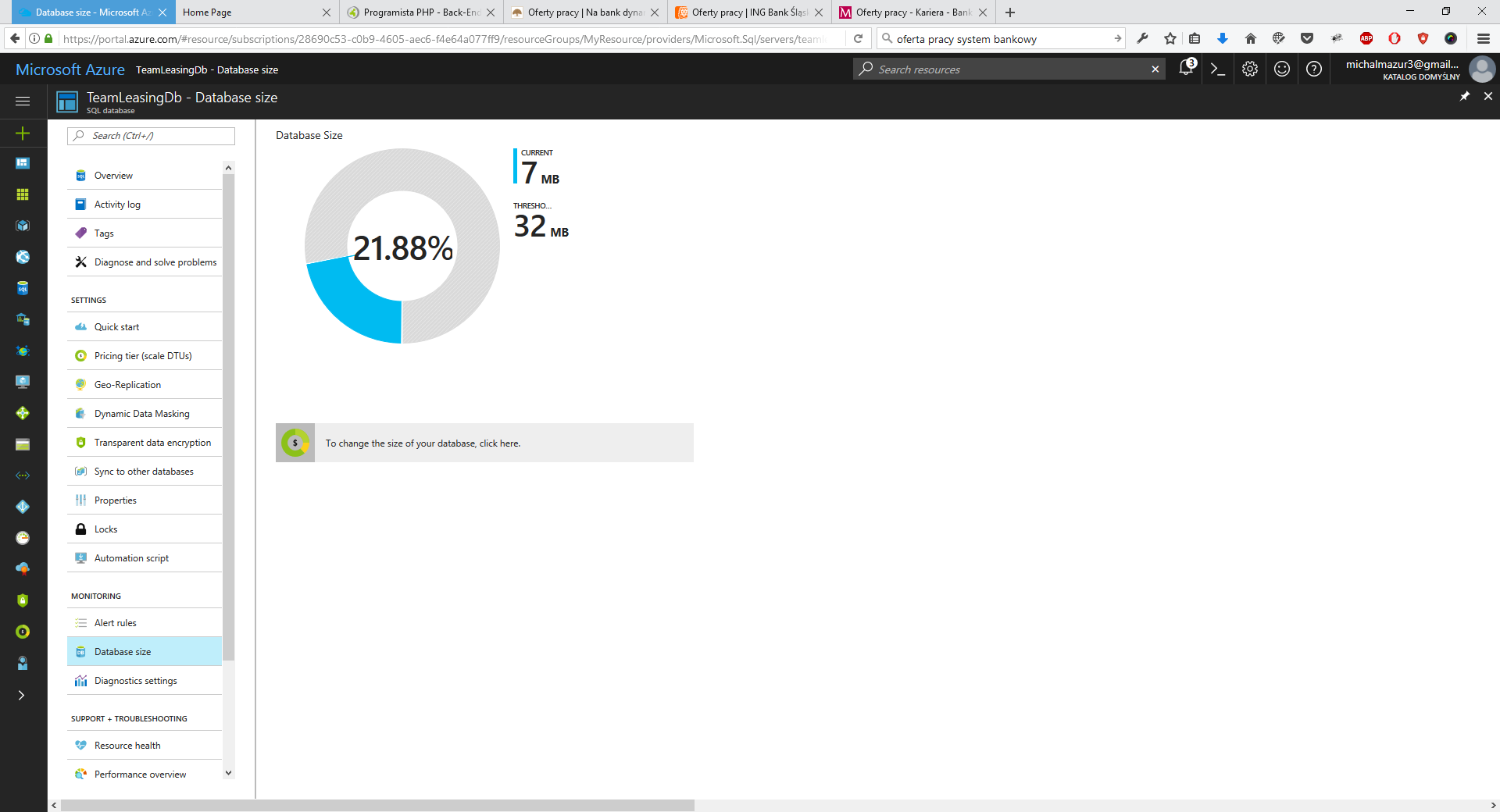
;

}

Inicjalizacji bazy danych przebiegła pomyślnie co można również zaobserwować w panelu, na portalu Azure. Jak widać na *Rysunek 8.6* oraz *Rysunek 8.7*, nastąpił mały ruch w naszym serwisie, wykres zużycia jednostek DTU wyraźnie się zwiększył, oraz miejsce w naszym magazynie uległo zmianie, dane inicjalizacyjne zajęły 7 MB z 35 MB jakie posiada nasz storage.



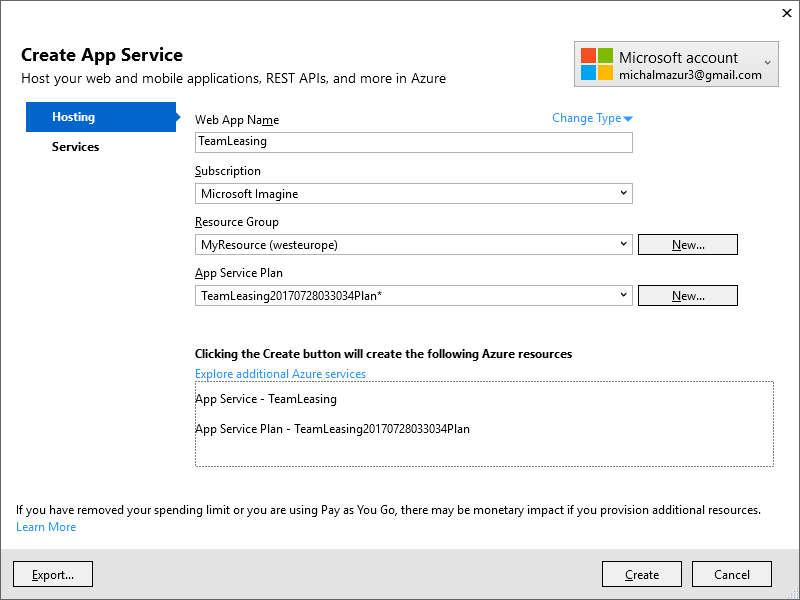
Rysunek 8.6 Wykres zużycia jednostek DTU



Rysunek 8.7 Wykorzystana powierzchnia magazynu

## Publikacja aplikacji

Kolejnym etapem jest przeniesie utworzonej aplikacji w obszar chmury obliczeniowej. W tym celu z poziomu Visual Studio 2017, należy we właściwościach solucji wybrać opcje „Publish”, a następnie określić podstawowe parametry takie jak nazwa aplikacji czy rodzaj subskrybcji (*Rysunek 8.8*).

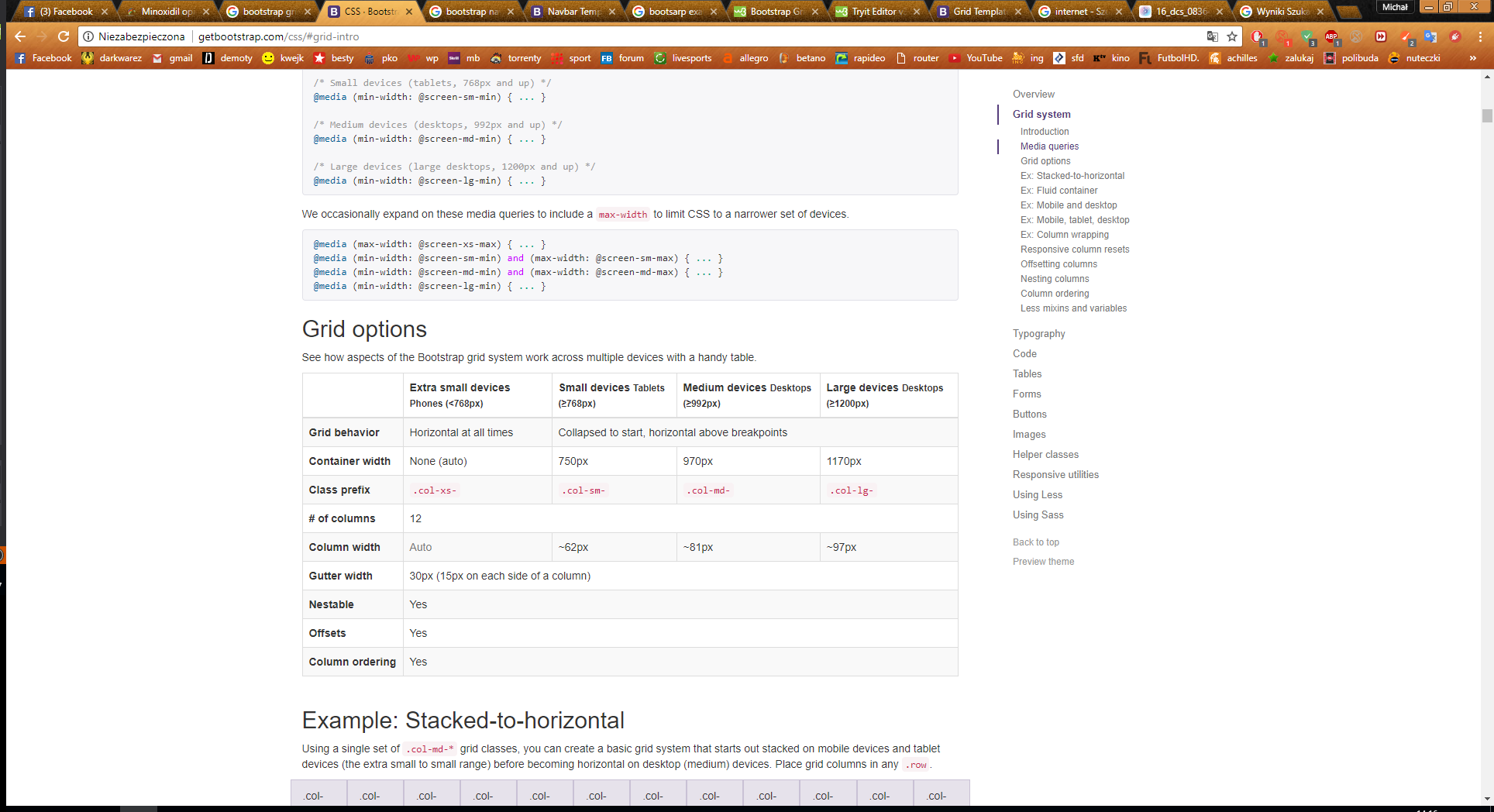


Rysunek 8.8 Publikacja aplikacji

# Testy aplikacji

## Testy na urządzeniach mobilnych

Do poprawnego wyświetlania serwisu, na urządzeniach mobilnych takich jak smartphon’y czy tablet’y zastosowano technikę Responsive Web Design. Do tego celo skorzystano z biblioteki Bootsrap, które oferuje takie możliwości jak określenie oddzielnego zestawu gridów (zestaw wierszy i kolumn) dla różnych rozdzielczości (Rysunek 9.1). W ten sposób łatwo można określić jak strona będzie się wyświetlała na różnych rozdzielczościach

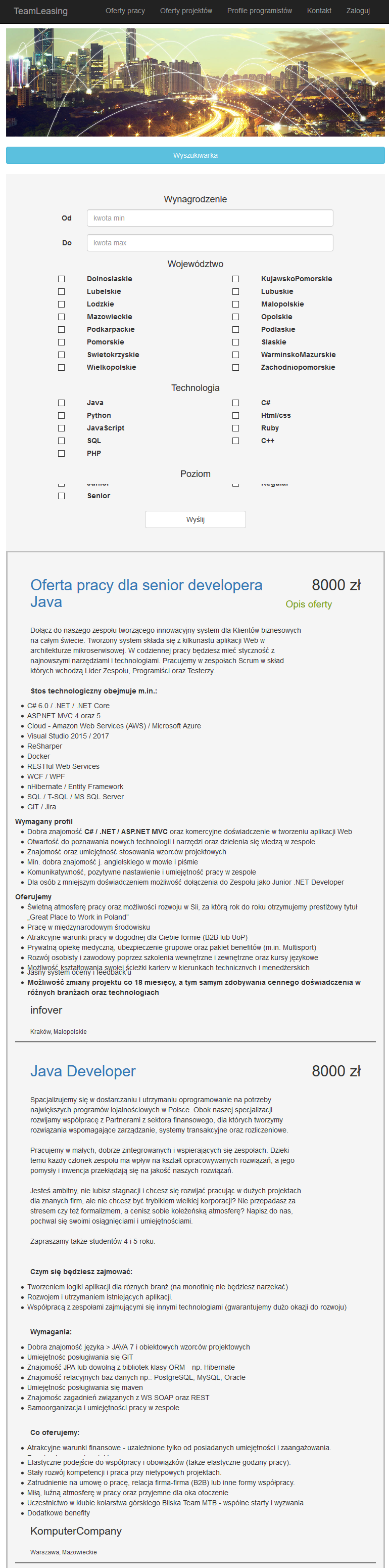


Rysunek 9.1 Obsługiwane rozdzielczości w bibliotece Bootsrap

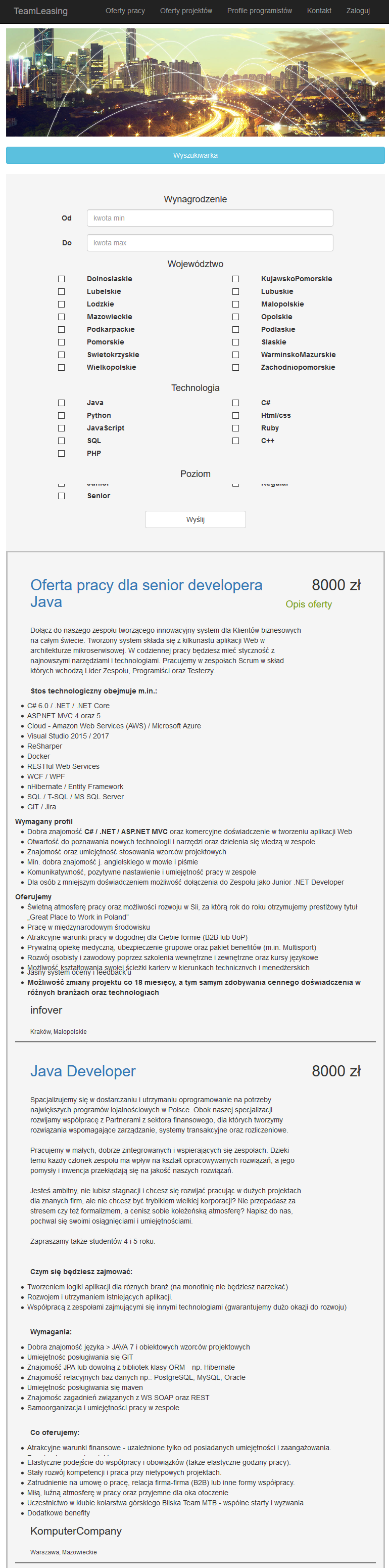
<http://getbootstrap.com/css/#grid-intro> dostęp z dnia 22.07.2017

### Tablet

Nie ma większego sensu, aby w pracy magisterskiej umieszczać zrzuty wszystkich podstron znajdujących się w serwisie, dlatego wybrano i umieszczono tylko dwie podstrony. Podstrona do wyszukiwania (*Rysunek 9.3*), jest dość skomplikowana i wymagała najwięcej czasu, aby ją poprawnie dostosować do wyświetlania na tabletach i smartfonach.W wersji na tablety i smartfony zawiera ona zwijalną wyszukiwarkę. Wyszukiwarka ta jest rozciągnięta ca całą szerokość, tak aby użytkownik korzystający z tabletu miał duży komfort podczas zaznaczania małych elementów takich jak checkbox’y (Rysunek 9.2).



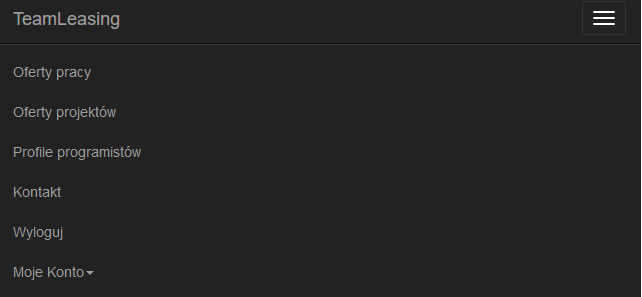
Rysunek 9.2 Wyszukiwarka w rozdzielczości na tablety



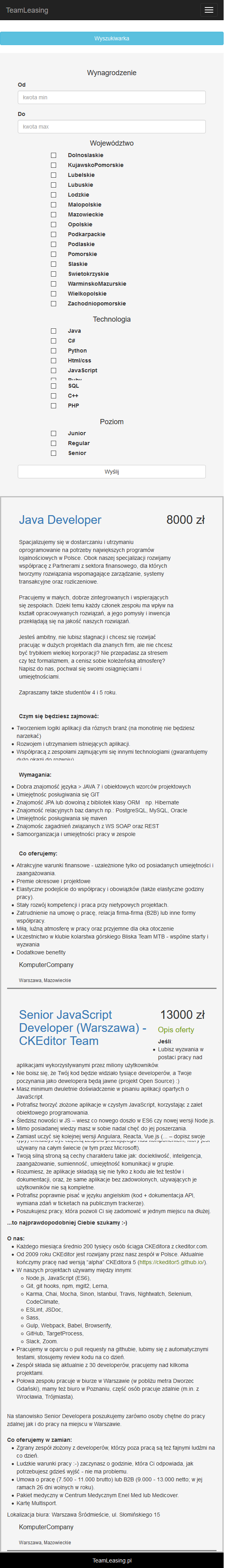
Rysunek 9.3 Długi zrzut ekranu dla podstrony wyszukiwania ofert pracy.

### Smartfon

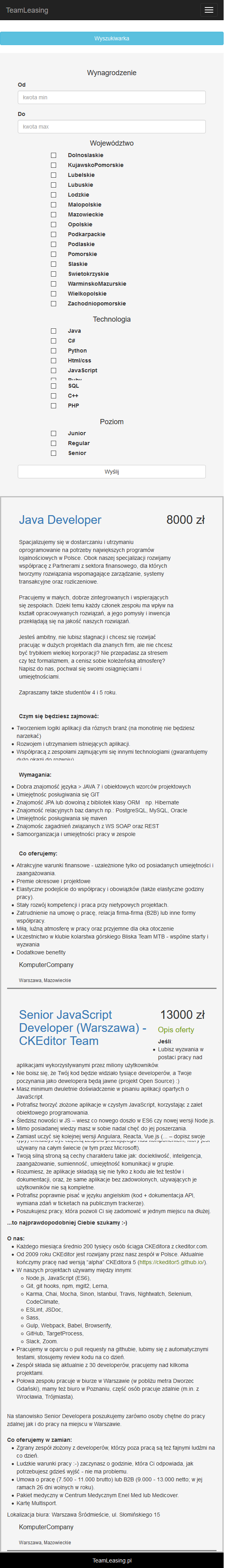
W rozdzielczości, którą posiadają współczesne telefony, główne menu ma postać wysuwanego menu znajdującego się pod przyciskiem (*Rysunek 9.4*). Dostosowano również wyszukiwarkę, w porównaniu do wersji na tablet jest tylko jedna kolumna z opcjami do zaznaczenia (*Rysunek 9.5*, *Rysunek 9.6*), oraz widok szczegółowy również pomniejszono i dostosowano do rozdzielczości jakie oferują smartfon’y (*Rysunek 9.7*)



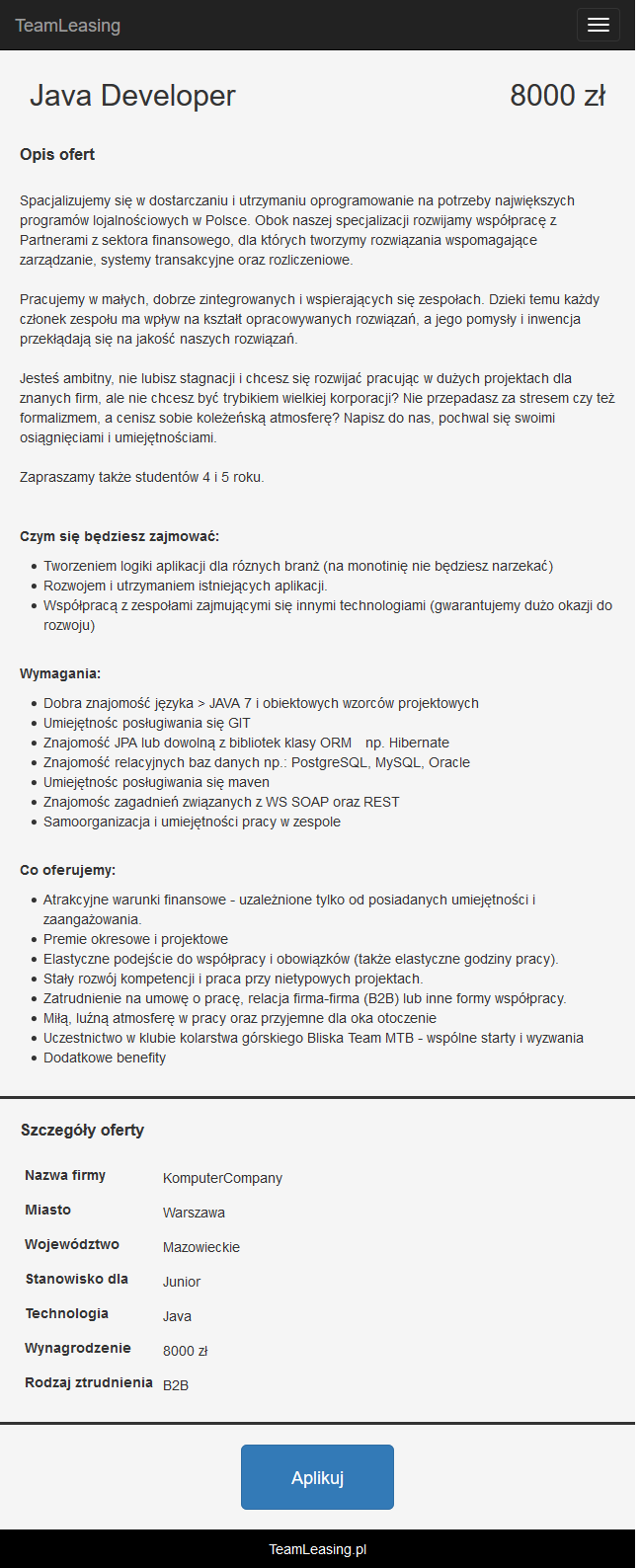
Rysunek 9.4 Główne menu – w wersji na telefon



Rysunek 9.5 Wyszukiwarka – wersji na telefon



Rysunek 9.6 Wyszukiwanie ofert pracy, długi zrzut – wersja na telefon



Rysunek 9.7 Widok szczegółowy oferty pracy – wersja na telefon

# Bibliografia

# [1] Główny Urząd Statystyczny <http://stat.gov.pl/metainformacje/slownik-pojec/pojecia-stosowane-w-statystyce-publicznej/3086,pojecie.html> dostęp z dnia 06.05.2017

# [2] The Cloud at Your Service: The When, How, and Why of Enterprise Cloud Computing

[3] <http://it.pwn.pl/Artykuly/Cloud-Computing/Cloud-computing-dla-niewtajemniczonych>

[4] <https://www.vmware.com/pl/company/news/releases/2011/2011-04-28-MSP.html>

# [5] [Marek Serafin](http://helion.pl/autorzy/marek-serafin) Wirtualizacja w praktyce Helion, Gliwice 2012

[6] Don Revelle, Hypervisors and Virtual Machines <https://www.usenix.org/system/files/login/articles/105498-Revelle.pdf>

[7] [Thomas Erl](https://www.google.pl/search?sa=X&biw=1366&bih=638&q=thomas+erl&stick=H4sIAAAAAAAAAOPgE-LRT9c3zEg3ycorSDZV4tLP1TdIrrI0NjfXkslOttJPys_P1i8vyiwpSc2LL88vyrZKLC3JyC8CACQ457s6AAAA&ved=0ahUKEwjd9Lur0d3TAhUJjiwKHf-6CiAQmxMItAIoATAO), [Zaigham Mahmood](https://www.google.pl/search?sa=X&biw=1366&bih=638&q=cloud+computing:+concepts,+technology+%26+architecture+zaigham+mahmood&stick=H4sIAAAAAAAAAOPgE-LRT9c3zEg3ycorSDZV4gXxDJNzk8sq87LNtGSyk630k_Lzs_XLizJLSlLz4svzi7KtEktLMvKLAEWJSHk9AAAA&ved=0ahUKEwjd9Lur0d3TAhUJjiwKHf-6CiAQmxMItQIoAjAO), [Ricardo Puttini](https://www.google.pl/search?sa=X&biw=1366&bih=638&q=ricardo+puttini&stick=H4sIAAAAAAAAAOPgE-LRT9c3zEg3ycorSDZVgvEyssqzK7RkspOt9JPy87P1y4syS0pS8-LL84uyrRJLSzLyiwB0xPd5PAAAAA&ved=0ahUKEwjd9Lur0d3TAhUJjiwKHf-6CiAQmxMItgIoAzAO) Cloud Computing: Concepts, Technology & Architecture 2013

[8] Peter Mell Timothy Grance, The NIST Definition of Cloud Computing, Computer Security Division Information Technology Laboratory National Institute of Standards and Technology, 2011

# [9] <http://www.inforisktoday.com/5-essential-characteristics-cloud-computing-a-4189> dostęp z dnia 06.05.2017

[10] <http://www.datacenterknowledge.com/archives/2014/10/13/explaining-community-cloud/>

[11 ]Srinivasan Sundra Rajan, *The importance of community clouds*, 24.04. 2011 r., http://cloudcomputing.syscon.com/node/1803698

[12] **Redakcja:** [prof. n. dr hab. Grażyna Szpor](https://www.ksiegarnia.beck.pl/autorzy/grazyna-szpor) INTERNET. Cloud computing. Przetwarzanie w chmurach, [C.H. Beck](https://www.profinfo.pl/f/c-h-beck:15,e,6/) , 2013

# [13] [Thomas Erl](https://www.amazon.com/Thomas-Erl/e/B001IGOVVY/ref=dp_byline_cont_book_1)  , [Ricardo Puttini](https://www.amazon.com/Ricardo-Puttini/e/B00B29RIPS/ref=dp_byline_cont_book_2)  , [Zaigham Mahmood](https://www.amazon.com/Zaigham-Mahmood/e/B00B29OIK6/ref=dp_byline_cont_book_3), Cloud Computing: Concepts, Technology & Architecture, Prentice Hall, 2013

# [14] [Barrie Sosinsky](https://www.google.pl/search?hl=pl&tbo=p&tbm=bks&q=inauthor:%22Barrie+Sosinsky%22), Cloud Computing Bible, Wiley Publishing 2011 [https://arpitapatel.files.wordpress.com/2014/10/cloud-computing-bible1.pdf dostęp z dnia 8.05.2017](https://arpitapatel.files.wordpress.com/2014/10/cloud-computing-bible1.pdf%20dostęp%20z%20dnia%208.05.2017)

# [15] <https://www.ibm.com/blogs/cloud-computing/2014/02/top-7-most-common-uses-of-cloud-computing/> dostęp z dnia 08.05.2017

# [16] <Http://www.pcworld.pl/news/Gry-webowe-i-rozwiazania-do-gier-w-chmurze,374754.html> dostęp z dnia 09.05.2017

# [17] [ADAM FREEMAN](https://www.google.pl/search?hl=pl&tbo=p&tbm=bks&q=inauthor:%22ADAM+FREEMAN%22), Pro ASP.NET Core MVC, **Springer-Verlag Berlin and Heidelberg Gmbh &, 2016**

[18]<https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/> dostęp z dnia 10.06.2017

[19] [https://github.com/aspnet/EntityFramework/wiki/Roadmap dostęp z dnia 11.06.2017](https://github.com/aspnet/EntityFramework/wiki/Roadmap%20dostęp%20z%20dnia%2011.06.2017)

[20] [Ben Dewey](https://www.google.pl/search?sa=X&biw=1366&bih=638&q=building+web+applications+with+visual+studio+2017:+using+.net+core+and+modern+javascript+frameworks+ben+dewey&stick=H4sIAAAAAAAAAOPgE-LVT9c3NEw2rSpOTk4xVIJyzQzMKgxNzbVkspOt9JPy87P1y4syS0pS8-LL84uyrRJLSzLyiwAsMPabPgAAAA&ved=0ahUKEwjHvNqtn7vUAhVEAZoKHbXpAj4QmxMIgQEoATAN), [Philip Japikse](https://www.google.pl/search?sa=X&biw=1366&bih=638&q=building+web+applications+with+visual+studio+2017:+using+.net+core+and+modern+javascript+frameworks+philip+japikse&stick=H4sIAAAAAAAAAOPgE-LVT9c3NEw2rSpOTk4xVIJwk8wM0kyzKzO0ZLKTrfST8vOz9cuLMktKUvPiy_OLsq0SS0sy8osACz88Mj4AAAA&ved=0ahUKEwjHvNqtn7vUAhVEAZoKHbXpAj4QmxMIggEoAjAN), [Kevin Grossnicklaus](https://www.google.pl/search?sa=X&biw=1366&bih=638&q=building+web+applications+with+visual+studio+2017:+using+.net+core+and+modern+javascript+frameworks+kevin+grossnicklaus&stick=H4sIAAAAAAAAAOPgE-LVT9c3NEw2rSpOTk4xVOIBc3PLis0Ny9O1ZLKTrfST8vOz9cuLMktKUvPiy_OLsq0SS0sy8osAck3k2z0AAAA&ved=0ahUKEwjHvNqtn7vUAhVEAZoKHbXpAj4QmxMIgwEoAzAN), Building Web Applications with Visual Studio 2017, Appress, 2017

[21] Krzysztof Żydzik, Tomasz Rak, Tworzenie nowoczesnych portali internetowych, Helion, 2015

[22] <http://edu.pjwstk.edu.pl/wyklady/tbo/scb/lecture-07/lecture-07.html>, dostęp z dnia 23.06.2017

[23] <https://www.tinymce.com/>, dostęp z dnia 11.07.2017

[24] <https://docs.microsoft.com/pl-pl/azure/sql-database/sql-database-what-is-a-dtu>, dostęp z dnia 23.07.2017